Pantelis Karapanagiotis | Marius Liebald

Entity Matching with Similarity Encoding: A Supervised Learning Recommendation Framework for Linking (Big) Data

SAFE Working Paper No. 398 | August 2023

Entity Matching with Similarity Encoding: A Supervised Learning Recommendation Framework for Linking (Big) Data[*]

Pantelis Karapanagiotis[†] Marius Liebald[‡]

This Version: August 15, 2023

Abstract

In this study, we introduce a novel entity matching (EM) framework. It combines state-of-the-art EM approaches based on Artificial Neural Networks (ANN) with a new similarity encoding derived from matching techniques that are prevalent in finance and economics. Our framework is on-par or outperforms alternative end-to-end frameworks in standard benchmark cases. Because similarity encoding is constructed using (edit) distances instead of semantic similarities, it avoids out-of-vocabulary problems when matching dirty data. We highlight this property by applying an EM application to dirty financial firm-level data extracted from historical archives.

Keywords: Entity matching, Entity resolution, Database linking, Machine learning, Record resolution, Similarity encoding
JEL Classification Code(s): C8

1. Introduction

Quantitative studies in finance and economics commonly rely on a multitude of data sources. However, they often lack a common identifier and matching their records requires a significant allocation of human and computational resources. In the age of big data, as the scale of the used sources grows, matching becomes less feasible because the costs of linking sources without common identifiers increase rapidly.

The recent literature on Entity Matching (EM) in finance, economic history, medical research, and computer science makes significant efforts in developing semi-automated

[*]We thank the participants of the *XIX World Economic History Congress* in Paris and the 1st *OCR Extraction Workshop* in Groningen for their constructive comments and suggestions. In particular, we want to thank Thierry Paquet and Bertrand Coüasnon as well as Uwe Walz for the valuable feedback. We gratefully acknowledge research support from the Leibniz Institute for Financial Research SAFE.

[†]Faculty of Economics & Philosophy, EBS Business School, Rheingaustraße 1, 65375 Oestrich-Winkel, Germany. Email: pantelis.karapanagiotis@ebs.edu. Phone: +49 (611) 7102 1530.

[‡]Department of Microeconomics & Management, Goethe University Frankfurt, Theodor-W.-Adorno Platz 4, 60323 Frankfurt, Germany. Email: liebald@econ.uni-frankfurt.de. Phone: +49 (69) 798 34819.

frameworks that reduce the human-in-the-loop requirements and enhance access to EM-driven research (Helgertz et al., 2022; P. Li et al., 2021; Abramitzky et al., 2020; López-Cuadrado et al., 2020; Antoni & Schnell, 2019; González-Carrasco et al., 2019; Ebraheem et al., 2018; Mudgal et al., 2018; Rodriguez-Lujan & Huerta, 2016). Nevertheless, these frameworks are accompanied by high technical burdens, and their implementation requires either fitting a comprehensive toolbox of Machine Learning (ML) models and selecting the most accurate (e.g., López-Cuadrado et al., 2020; González-Carrasco et al., 2019) or selecting among vocabularies and using word embeddings to fit an appropriate language model (e.g., López-Cuadrado et al., 2020; Ebraheem et al., 2018).

We propose a modular EM recommendation framework based on Artificial Neural Networks (ANN) that, similarly to other state-of-the-art systems, aims to reduce human-in-the-loop requirements while keeping the technical prerequisites within the set of well-established methods used in finance and economics. Because ML methods are not compatible with alphanumeric data, a common feature of the frameworks based on such methods is the presence of an initial encoding step transforming text to numeric data. Alternative frameworks calculate the similarity of two records by first embedding the textual data of each record into two distinct vector representations and then calculating the similarity of the vector representations. Instead, our approach bypasses the separate embedding of records and introduces a similarity encoder that directly transforms a pair of records to ML-compatible numeric data using distance concepts (e.g., Levenshtein, Jaro-Winkler, Euclidean, etc.) that are ubiquitous in economic and finance on EM.

Our approach has the following advantages: (1) Similar to state-of-the-art models, it reduces the need for human expertise on the content of the linked data sources. (2) By avoiding word embeddings, it requires less technical expertise than other matching frameworks. (3) The similarity encoder, without excluding the possibility, removes the need to train a model using word-embedding layers that can be computationally costly. (4) The similarity encoding calculations for each record pair are independent of each other and can be parallelized. As a result, the approach we propose can be efficiently scaled to multiple processing units and nodes to handle matching cases with a large number of records. (5) Because similarity encoding gives similarity representations of records based on (edit) distances, it is not susceptible to Out-of-Vocabulary problems that arise when using word embeddings from multilingual and/or dirty data (e.g., data containing typos, spelling mistakes, or extraction errors) (Piktus et al., 2019).

Besides using semi-automated frameworks, record matching via human computation is prevalent in economics, finance, and, more generally, in social sciences (e.g., Bartram et al., 2022; Wojcik et al., 2021; Persson, 2020). However, human computation is costly and can even become unfeasible when working with large-scale, administrative, online tracking, and other forms of big data. For such cases, human computation is only feasible when the linking procedure can be crowd-sourced, which in turn requires (i) the absence of privacy restrictions that prevent disseminating the data to the public, and (ii) the availability of either an established community willing to engage in the process or enough financial resources to fund the record linking labor. These requirements can prevent independent researchers and academic institutions with limited access to funding from conducting research using linked big data, give competitive advantages to researchers and institutions based on financial instead of academic merits, and hinder the democratization of EM-related research (Ebraheem et al., 2018). Recent end-to-end frameworks provide semi-automated recommendation systems using model and feature selections; blocking and matching-threshold rules; and, more recently, ANNs aiming to reduce the allocation of human resources to EM tasks (P. Li et al., 2021; Antoni & Schnell, 2019; Ebraheem et al., 2018; Mudgal et al., 2018).

We demonstrate that our recommendation framework is on-par or outperforms state-of-the-art end-to-end alternatives when detecting record matches with minimal human input. We benchmark the accuracy of our approach using three matching tasks with publicly available data sets that are commonly used in evaluating EM frameworks.[1] Although our framework is highly modular and can be used in combination with blocking layers, the few-shot learning property of the ANN underlying our framework allows the benchmark results to be obtained within reasonable computation times (few hours) without resorting to blocking rules (see also P. Li et al., 2021; Mudgal et al., 2018, for other Deep Learning (DL) models with similar properties).

Finally, we present an application of our matching framework in a domain with dirty firm-level financial data that we extracted from historical archives by using Optical Character Recognition (OCR) software (Kamlah et al., 2022). The data represent German firms operating in the period from 1910 to 1919 with non-harmonized and non-standardized attributes extracted from the "Handbuch der deutschen Aktiengesellschaften" (see also Gram et al., 2022). In a 5-fold cross-validation with 30% train and 70% test random sample splits, our framework achieves an average 99.36 F-score in the test sub-sample. Our results strongly indicate that record matching with similarity encoding provides a lightweight alternative that gives high-accuracy recommendations when linking dirty data.

The rest of the paper is organized as follows. Section 2 contains a summary of the previous work on EM with particular emphasis on frameworks and applications in finance and economics. In Section 3, we introduce the ANN architecture of our EM approach and discusses similarity encoding from (i) a system design and (ii) a formal, and (iii) a data transformation perspective. In Section 4, we discuss the modularity of similarity encoding layers and the computational properties of our implementation. In Section 5, we present the evaluation of our framework against several benchmark EM tasks found in the

[1]The benchmark tasks match entities between (i) *Abt-Buy* (E-commerce), (ii) *Amazon-GoogleProducts* (E-commerce), and *DBLP-ACM* (bibliographic) data.

literature and compare it with the evaluations of alternative state-of-the-art frameworks. Furthermore, the section presents the results of the application of our framework when linking dirty financial historical data. Finally, Section 6 concludes.

2. Literature Review

2.1 Entity Matching Practices in Finance and Economics

EM (also referred to as record or entity linkage, and entity resolution) is the process of linking records from distinct data sources. In this context, an *entity* can refer to a multitude of objects, such as corporate organizations, individuals, municipal bodies as well as digital entities, such as social media accounts or research articles.

EM is an important part of (quantitative) research in the fields of economics and finance. This importance is because – assuming the researchers combine different sources – it is a necessary step in generating a study's data foundation. The EM frameworks that are dedicated to and designed from within the social sciences primarily focus on the linking of individuals across historical sources. In particular, social scientists have focused on linking US census data and have developed automated EM systems. While the early approaches used hardcoded matching rules introduced by Ferrie (1996) and Ferrie (2005), the more recent literature has proposed linkage frameworks that rely on statistical algorithms. These frameworks use expectation maximization (Abramitzky et al., 2020) or supervised learning techniques. When using ML to find matching record pairs, the proposed frameworks rely on logit (Helgertz et al., 2022) and probit (Feigenbaum, 2016) regressions or gradient boosting (Price et al., 2021). Bailey et al. (2020), Abramitzky, Boustan, Eriksson, et al. (2021) and Ghosh et al. (2023) evaluate these frameworks' accuracy and find somewhat heterogeneous performances. Importantly, ML-based frameworks making use of deep ANNs are still absent from this literature to the best of our knowledge.

Next, we discuss the EM approaches used in quantitative studies in economics and finance in cases where a common identifier does not exist. Reviewing the literature indicates the existence of three essential approaches. These include (i) human computation-based EM (e.g., Florackis et al., 2023; Bartram et al., 2022; Persson, 2020), (ii) rule-based approaches that refer to manually defined thresholds for record pair similarities, and (iii) applications of the automated frameworks discussed in the previous paragraph (e.g., Buggle et al., 2023; Abramitzky, Boustan, Jacome, & Perez, 2021).

To gain an impression about the relative distribution of these three approaches, we investigated EM practices in articles published in two recent issues of *The Journal of Finance* (Vol. 78, Issue 3) and *The Quarterly Journal of Economics* (Vol. 138, Issue 2).[2] First, as illustrated by Table 4 in Appendix D, 11 of the 24 articles published in these two

[2]We select these two journals as a reference because they are the highest ranked outlets in their fields according to Scimago.

issues are based on EM tasks, highlighting the relevance of the topic. Second, only one articles applies EM based on the statistical algorithms discussed above. The remaining articles either use rule-based methods that rely on manually defined thresholds for record pair similarities or perform human computation-based EM.

In summary, researchers in finance and economics implement EM in different ways. However, EM based on ANN appears to be missing. This is surprising considering that such DL-based approaches yield the best results (see Section 5) and represent the state-of-the-art in computer science (see Section 2.2).

2.2 Entity Matching in Computer Science

From a computer science perspective, EM is crucial for information management and analysis as well as knowledge integration and data mining. Although the research community had already acknowledged the importance of EM more than 75 years ago (e.g., Dunn, 1946), most of the development took place during the last two decades.

The methods proposed encompass rule-based (Singh et al., 2017; Fan et al., 2009; Bilenko & Mooney, 2003; J. Wang et al., 2011)[3] and crowdsourcing (Gokhale et al., 2014; J. Wang et al., 2012) approaches. More recently, however, ML (Qian et al., 2017; Konda et al., 2016; Bilenko & Mooney, 2003) and in particular DL-based methods have received greater attention as they exhibit superior performances (see, for instance, Section 5).

The DL-based approaches primarily work on information provided by external word embedding models. Differences stem from the type of embedding models used. On the one hand, EM systems, such as Y. Li et al. (2020)'s DITTO, leverage pre-trained transfomer-based language models (LM) that enable *contextual* word embeddings to account for the different meanings of a given word. On the other hand, DEEPER (Ebraheem et al., 2018), DEEPMATCHER (Mudgal et al., 2018), and CORDEL (Z. Wang et al., 2020) represent examples for EM systems that rely on embedding models that do not use self-attention mechanisms. Commonly used word embedding models in this context are fastText (Bojanowski et al., 2017) or GloVe (Pennington et al., 2014).[4] Moreover, efforts are being undertaken that aim to reduce the costs associated with generating sufficient training data for DL-based methods, including active learning (Meduri et al., 2020; Kasai et al., 2019; Qian et al., 2017), data augmentation (Y. Li et al., 2020), transfer learning techniques (Jin et al., 2021; Zhao & He, 2019), or unsupervised ML approaches such as AUTOFJ (P. Li et al., 2021) and ZEROER (Wu et al., 2020).

A conventional EM pipeline typically consists of four steps (Ebraheem et al., 2018):

[3]Rule-based approaches used in finance and economics do not correspond to recent computer science work related to rule-based EM. By contrast, while the default approach in finance and economics rests on manually defined similarity thresholds, the computer sciences focus on the automatic generation of such rules without human involvement (e.g., Singh et al., 2017).

[4]See Y. Li et al. (2020) for a more detailed literature review on DL-based EM approaches that include a discussion of the field's challenges and opportunities.

(1) creating a labeled training dataset, consisting of true and false matching pairs, (2) training a ML model or learning decision rules using (a subset) of the labeled dataset, (3) *blocking* or other indexing techniques that aim at minimizing computational and memory requirements by reducing the pool of candidate record pairs, and (4) making predictions using the learned ML model and rules from (2).

3. Similarity Encoding

3.1 An Intuitive Description of the Framework

We want to match the records of two data sets, say D_l and D_r, that have some overlapping information. The records of D_l and D_r have one (e.g., unstructured data) or more (structured data) fields. A subset of the fields of D_l and D_r has values drawn from the same information space.

For example, D_l and D_r are two structured data sets containing records of automobiles. The data set D_l has the fields *model*, *producer*, and *origin*, which take alphanumeric values and a field *sales* with numeric values. The data set D_r has the fields *name*, *firm*, and *country* with alphanumeric and *engine* with numeric values. Despite the different naming conventions, the fields *model* and *name* contain non-harmonized automobile model names, *producer* and *firm* have non-harmonized producing firm names, and *origin* and *country* have the country of origin of the producing firm.

A straightforward manual attempt to find the records from D_l and D_r that belong to the same entity involves pairing the associated fields by drawing values from the same information space and examining how similar the information they represent is across all records. The advantage of this approach is that it incorporates information coming from multiple fields that humans can use to make their assessments. The humans consolidate the isolated field similarities in an overarching record similarity evaluation. However, the disadvantage of this manual approach is, however, that as D_l and D_r become larger, any manual EM endeavor becomes costly and time inefficient.

The EM framework we introduce uses a collection of field and record ANNs to emulate manual EM behavior. Figure 1 displays the architecture.[5] The ANN architecture imitates the previously described human matching process while being inexpensive and far less time-consuming. An overarching record-matching network distributes similarity data to separate small field networks. Each field network outputs a field prediction (probability) that indicates whether two values from a pair of D_l and D_r fields constitute a match. Subsequently, the field predictions are concatenated and passed through the overarching record network. The record network consolidates the separate field predictions and outputs the probability that a pair of D_l and D_r records represent the same entity.

[5]We postpone the discussion of the technical advantages of the introduced architecture for Section 4 and provide a high-level framework description in this section.

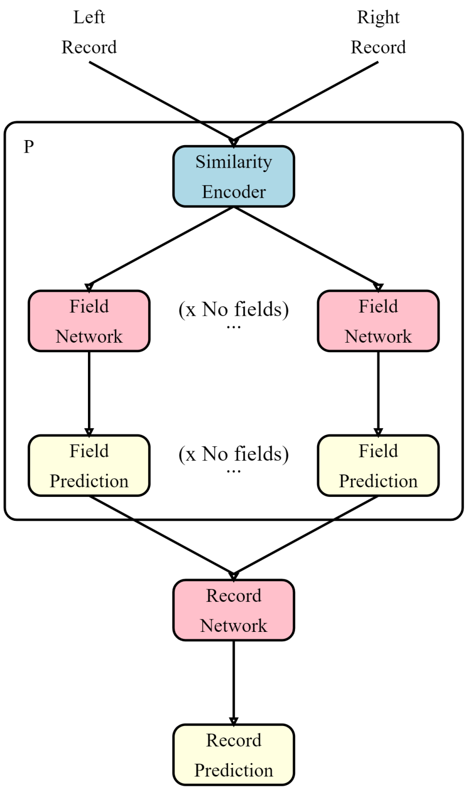

Figure 1: Entity matching network architecture.

A crucial element that is missing from the discussion so far is how each field network obtains the field similarity values. In most cases, the similarity of the values from the D_l and D_r fields can be calculated in many distinct ways. For instance, the similarity of two alphanumeric fields can be calculated via non-equivalent variations of the edit distance such as the Levenshtein (Levenshtein, 1965) and Hamming (Hamming, 1950) metrics. As an example, the Levenshtein and Hamming metrics with unit cost operations of the values "Acme Corporation" and "Corporation Acme" are respectively equal to 10 and 11. Furthermore, there are alternative approaches for assessing the similarity of two alphanumeric values that are not based on (mathematically) proper metrics. For example, one can tokenize the two sentences of the last example, order the tokens, and then calculate an edit distance between the transformed sentences.[6] This distance calculation results in comparing the transformed expressions "Acme Corporation" and "Acme Corporation" which are identical. As a result, some concepts of similarity can be more appropriate than others for certain pairs of field values.

When humans manually assess the similarity of two D_l and D_r fields, they are not confined to using a single calculation method. Recognizing that "Acme Corporation" and

[6]This approach is commonly referred to as token sort ratio. Tokenization describes splitting a string into parts (i.e., tokens) using a special character as the token delimiter (usually a space character).

"Corporation Acme" are identical up to a reshuffling transformation is an approachable realization for many humans examining the similarity of the two values. A human can also quickly realize that the reshuffling technique is irrelevant when comparing the values "Acme Germany" and "Acme Greece" and resort to alternative methods for assessing the similarity of the fields. The component of the matching architecture in Figure 1 responsible for emulating the use of multiple distance concepts is the similarity encoder.

The similarity encoder receives as input a pair of records from D_l and D_r and performs a collection of similarity calculations for each pair of associated fields. One or more similarity calculation methods can be used for each pair of associated fields. The architecture displayed in Figure 1 disentangles the encoding of input records to similarity values from the evaluation of the probability of the input records' entities constituting a match. The field and record networks are responsible for assessing the matching probability, while the similarity encoder is responsible for transforming the input data to numerical values that can be used in statistical learning methods.

The advantage of the separation is that the user neither needs to specify blocking rules and thresholds nor evaluate constraints regarding the appropriateness of individual similarity calculations. Instead, the statistical learning method, in our case the field and record networks, assume this evaluation. It suffices that the user provides high-level instructions to the similarity encoder on which (i) fields to use from each data set, and (ii) which similarity calculation methods are potentially relevant for each pair of associated fields.

3.2 A Formalization of Similarity Encoding

Rozinek & Mareš (2021) provide an axiomatization of similarity spaces extending the usual concepts used for metric spaces and show that similarity and metric spaces have a duality relation based on negative monotonic transactions. We extend certain aspects of their work providing new results that are relevant for EM with similarity encoding. Intuitively, we show that whenever two or more fields of two data sets contain elements from spaces that comply with a concept of distance, they can be encoded via similarity functions to vector spaces, which are compatible with statistical learning methods.

A normalized similarity space is a pair (X, s) where X is a set and $s\colon X \times X \to [0, 1]$ is a symmetric function that satisfies $s(x, y) = 1 \iff x = y$ and $s(x, z) + 1 \geq s(x, y) + s(y, z)$ for all $x, y, z \in X$. Our formalization is driven by Proposition 1.

Proposition 1 *A set X is metrizable if and only if there is a normalized similarity s making (X, s) a normalized similarity space.*

The proof of Proposition 1 is given in Appendix A. The relevance of Proposition 1 is better illustrated by focusing on examples that are commonly found in EM applications.

1. **Fields with numeric values:** Any two numeric fields that can be embedded in the real or complex number system admit normalized similarities. For example, fields having integer values can be embedded in the real number system, and the function $s(x, y) = 1/ (|x - y| + 1)$ can be used as a normalized similarity. In particular, fields drawing values from normed vector spaces have normalized similarities. For example, any ℓ_p space becomes a normalized similarity space when coupled with the function $s(x, y) = 1/ (\|x - y\|_p + 1)$.

2. **Fields with date or time values:** The similarity of date and time fields can be assessed in terms of time units (e.g., seconds, minutes, and hours) that have passed since a fixed reference point. Practically, this is a special case of the previous example. The most common date or time representation used in computing is the UNIX time, measuring the number of seconds elapsed since midnight UTC of January 1, 1970 (UNIX epoch). With this representation, the similarity between two dates is reduced to the similarity of the numeric values of their UNIX times.

3. **Fields with geographical coordinates:** There are many ways with various degrees of accuracy for calculating distances on the earth's surface (see, e.g., Stal et al., 2022). Any reasonable distance d of two points on the surface of the earth is naturally bound by half of the earth's circumference. For applications with coordinates confined within a particular geographical region the upper bound of the distance d is $M = \max_{x,y} d(x, y)$ where x and y are picked from the coordinates belonging in this region. A normalized similarity can then be obtained by $s(x, y) = (M - d(x, y))/M$.

4. **Fields with alphanumeric values:** Edit distances with unit operation costs and invertible operations are metrics for a given alphabet over which the strings of the two associated fields are formulated. Fisman et al. (2022) show that the normalized edit distance (NED; Marzal & Vidal, 1993) with unit operation costs is a metric. A normalized similarity can be obtained from NED by $s(x, y) = 1 - d(x, y)$.

Many functions used in practical applications to measure the similarity of the values from two fields cannot be formally induced by a concept of distance. For example, the Jaro-Winkler distance, which is a commonly used approach in measuring how similar two strings are in record-matching applications (Adam et al., 2021; Cule et al., 2020), does not satisfy the triangular inequality condition (Rozinek & Mareš, 2021, Definition 4, Property N2). In such cases, the theoretical guarantee of Proposition 1 and some results of Fisman et al. (2022); Rozinek & Mareš (2021) are not applicable. Nonetheless, functions such as the Jaro-Winkler distance have established their usefulness in practical applications and, as long as the user tolerates not having the theoretical guarantees on some occasions (see Section 5.2), non-proper similarity functions can be used with the network architecture displayed in Figure 1.

The remaining section focuses on the construction of similarity encoders from collections of field similarities. Let $X_1, \ldots, X_k$ be sets from which the values of the common fields of the data corpora are drawn. Let also $X_{l,1}, \ldots, X_{l,k_l}$ and $X_{r,1}, \ldots, X_{r,k_r}$ be the potentially distinct sets for the values of the remaining fields of the corpora to be matched. Consider the product spaces

$$Y_s = \left(\prod_{i=1}^{k} X_i\right) \times \left(\prod_{i=1}^{k_s} X_{s,i}\right) \quad (s = l, r)\,,$$

which we will refer to as **left space** ($s = l$) and **right space** ($s = r$). Let $D_l \in Y_l^{n_l}$ be a $n_l \times (k + k_l)$ and $D_r \in Y_r^{n_r}$ be a $n_r \times (k + k_r)$ matrix. Note that the matrices D_l and D_r, which represent the two data sets that we aim to link, can have different row numbers (i.e. $n_l \neq n_r$).

Suppose that the rows of the matrices D_l and D_r are associated by some application-specific logic, which can be described using field similarity functions over the common spaces $X_1, \ldots, X_k$. Let m_i be the number of similarity functions used for field X_i with $i = 1, \ldots, k$ and $m = \sum m_i$ be the total number of used similarity functions. In our framework's logic, these field similarity functions represent instructions on how two records of D_l and D_r can be encoded into data compatible with statistical learning methods.

Each distinct set of instructions based on the field constituents of the records gives rise to a distinct similarity encoder. For a fixed set of instructions

$$\mathcal{I} = \left(s_1^1, \ldots, s_{m_1}^1, s_1^2, \ldots, s_{m_2}^2, \ldots, s_1^k, \ldots, s_{m_k}^k\right)\,, \tag{1}$$

where s_j^i is the j-th field similarity applied to element i, we can define a vector-valued function by bundling the similarities together; namely

$$\begin{aligned} c_{\mathcal{I}} \colon & \left(\prod_{i=1}^{k} X_i\right) \times \left(\prod_{i=1}^{k} X_i\right) \to [0,1]^m \\ \colon & \left(x^l, x^r\right) \mapsto \left(s_1^1(x_1^l, x_1^r), \ldots, s_{m_1}^1(x_1^l, x_1^r), \ldots, s_1^k(x_k^l, x_k^r), \ldots, s_{m_k}^k(x_k^l, x_k^r))\right)\,, \end{aligned} \tag{2}$$

where x_i^s denotes the i-th element of the input vector x^s (for $s = l, r$). We refer to functions defined as in Equation 2 as **similarity encoders**. For any given instruction set, a similarity encoder describes how the two records of D_l and D_r are associated.

As noted earlier, there are typically multiple ways one can define associations between two data sets. For example, when matching two string fields both the Levenshtein and the longest common subsequence (LCS) similarities can be used. Thus, the similarity encoder defined by $\mathcal{I}$ is not unique, and some sets of instructions are more appropriate for certain application domains than others. Therefore, a flexible EM recommendation framework should allow switching between various instruction sets instead of attempting to provide a universal similarity encoder based on the (data) types of the associated fields.

To describe the mapping of instructions to similarity encoders, let $\mathcal{N}$ be the collection of all instructions with m_i similarity functions for field i that can be defined when matching the left and right spaces. A **similarity map** is a function that gives an encoder for each instruction set; and it is defined as

$$\begin{aligned} \mathrm{S} \colon \mathcal{N} &\to \left([0,1]^m\right)^{\left(\prod_{i=1}^k X_i\right)\times\left(\prod_{i=1}^k X_i\right)} \\ \colon \mathcal{N} &\ni \mathcal{I} \mapsto c_{\mathcal{I}}. \end{aligned} \tag{3}$$

In the architecture displayed in Figure 1, the operation of Equation 3 is implemented by the constructor of the similarity encoder (see also Section 4 and Appendix B for implementation details).

The similarity map creates a vector-valued function for each collection of real-valued functions passed to it. Because S is a special case of concatenation of scalar functions, it satisfies the following proposition.

Proposition 2 *With the product norm induced by the supremum norm used on its domain and the ℓ^1 norm on its range, the similarity map* S *is an isometry. In particular,* S *is a restriction of a bounded and continuous operator with operator norm equal to 1. Moreover,* S *maps convex combinations of instructions to convex combinations of similarity encoders.*

The proof of Proposition 2 is in Appendix A. The isometry property of Proposition 2 indicates that the similarity operator maps the individual distances of the field spaces X_i to similarity encoders that capture overarching distances between records. The arguments of the proof of Proposition 2 depend only on the ranges of the similarity functions which are $[0, 1]$ by definition. Therefore, they can be replicated for functions such as Jaro-Winkler that do not satisfy the definition of a similarity function (and hence Proposition 1). Nevertheless, the problem when attempting to expand the definition of S to include such cases is that there is no unambiguous way established in the literature of which type of functions should be considered as instructions. Thus, there is no established way to define a replacement for $\mathcal{N}$ which makes the definition of S over non-proper similarity functions ambiguous.

3.3 Similarity Encoding from a Data Transformation Perspective

A similarity encoder $c_{\mathcal{I}}$ receives input data in the form of associated field pairs from two data sets and outputs a similarity matrix. The field pairs of the input data can have different data types (e.g., alphanumeric, numeric, date, etc.). All output columns have numeric data types. Moreover, the number of columns in the output matrix can be greater than the number of field pairs when the instructions used to construct $c_{\mathcal{I}}$ contain more than one similarity function for any of the associated fields. The data transformation is displayed in Figure 2.

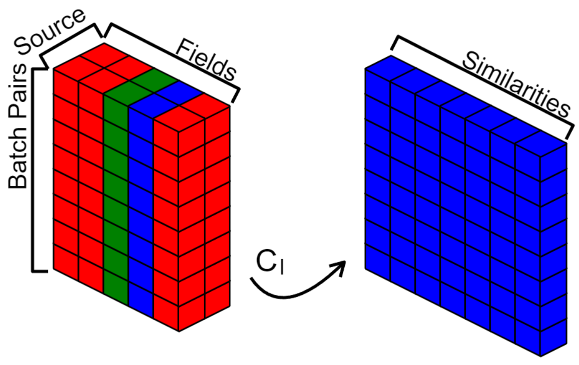

Figure 2: Similarity encoding transformation.

The input data have three dimensions (left rectangular cuboid of Figure 2). The rows represent a batch of record pairs obtained by combining rows from the left and right data sets, the columns represent the associated fields, and the depth represents the data source (left or right data set). The color filling of each column indicates the data type of the associated fields. Columns 1, 2, and 5 have the same data type (red; say alphanumeric), column 3 has a different data type (green; say date), and column 4 is of numeric type (blue).

The output data have two dimensions (right rectangular cuboid of Figure 2). The rows represent the batch of record pairs, and the columns represent the similarity values calculated by $c_{\mathcal{I}}$. The output matrix has seven columns, two more compared to the columns of the input data, implying that the encoder applies either three similarity functions to one of the field pairs, or two similarity functions to two field pairs. All columns of the output data have numeric values in the interval $[0, 1]$. The similarity matrix produced by the encoder is used as input for the subsequent layers (field networks) of the architecture displayed in Figure 1.

4. Implementation and Discussion

The two main challenges associated with EM boil down to computation time and memory requirements. Completely matching the entities represented by the records of two data corpora requires comparing all the pairs of the cross-product of the corpora. However, the number of pairs grows quadratically with the number of observations of the left and right data sets which makes the EM tasks computationally demanding even for small data sets (Christen, 2012). For example, if $n_l = n_r = 10^5$ are the numbers of left and right records respectively, there exist $n = 10^{10}$ candidate record pairs. Even if no more than one similarity function is used for only one pair of fields, this number of candidate pairs requires 40Gb of memory when storing single precision floating numbers (4 bytes).

4.1 Data- and Technical-expertise Requirements

Conventional end-to-end EM systems use blocking as a central component of their pipelines to mitigate the matching challenges. Blocking reduces computational and memory requirements by lowering the number of candidate record pairs based on application-specific sets of exclusion constraints. In this respect, blocking techniques introduce a trade-off between comparison completeness and computation time. In conventional systems, the exclusion constraints are manually defined by humans which requires existing expertise with the semantics, content, and accuracy of the data sources. In this respect, the trade-off between comparison completeness and computation time translates into a trade-off between data expertise and computation time.

However, data expertise is application specific and, due to the idiosyncrasies of entity matching problems, not easily transferable between applications even if these fall into the same data domain. Every matching problem requires finding appropriate blocking criteria, and there is no guarantee, for example, that the criteria for matching person entities between sources D_l and D_r are effective when matching person entities between sources D'_l and D'_r. Furthermore, the efficiency of the conventional blocking approach becomes more doubtful when the total cost of using all the resources, both human and artificial, is considered. Gaining data expertise is a time and effort-intensive task for humans, and the exchange of machine computation time with human resources can hinder certain applications.

Many state-of-the-art EM systems found in the literature propose statistical methods for selecting exclusion constraints that require involving less data-expertise at the cost of using more technical-expertise. For example, inclusion and exclusion criteria can be selected from the combinations of the fields using techniques such as Locality Sensitive Hashing (LSH) (e.g., Ebraheem et al., 2018). Nonetheless, even blocking implementations based on statistical learning can lead to an unintended exclusion of actual entity matches (e.g., due to statistical errors) that negatively affect the EM's performance. This risk can significantly increase if some data fields are dirty and the representation of the corresponding entity characteristics is imperfect.

Our framework, similar to other state-of-the-art EM systems (P. Li et al., 2021; Ebraheem et al., 2018; Mudgal et al., 2018), uses DL to reach matching recommendations. Matching models based on DL have a few-shot learning property in the sense that the models can learn how to detect record matches by using only a subset of the available match and mismatch examples (see also Mudgal et al., 2018). This property greatly reduces the number of record pairs required for model training and alleviates the need for blocking.[7] For example, instead of using all $n = n_l \times n_r$ candidate record pairs, model training can be based on n_m match examples and only a subset of the remaining $n - n_m$ mismatch examples. The default behavior in the implementation of our framework is to use 10% of the mismatch examples during training. The results for the benchmark evaluations and the application in Section 5 are obtained using the default 10% mismatch share.

4.2 Semantic and Distance Matching with Dirty Data

While most of the finance and economics literature relies on numeric and string distances to match data sources, many state-of-the-art systems outside these fields are based on word embedding techniques. The standard techniques give vector representations of words that capture semantic similarities, and EM systems use these semantic similarities to provide matching recommendations when linking data sources. However, many big data sources in finance and economics are multilingual and contain information from user input, which can have spelling mistakes, or OCR-extracted text, which may have recognition errors. Such cases lead to Out-of-Vocabulary problems and hinder the applicability of word embedding methods (Piktus et al., 2019).

A solution to the Out-of-Vocabulary problem is to increase the embedding space so it can represent more, potentially misspelled or OCR miss-recognized words. On the other hand, statistical learning using large vocabularies is computationally demanding. Instead, distance matching is based on numeric and edit distances, does not require establishing a vocabulary to calculate similarities, and is readily applicable for cases of dirty data.

The similarity encoding implementation we propose can simultaneously account for multiple similarity functions for a single pair of associated fields.[8] Because a similarity function's applicability is context-specific (Abramitzky et al., 2020), this flexibility is a central design feature of our framework. Every matching problem requires finding an appropriate similarity function, and there is no guarantee that, for example, the similarity function of choice for matching persons between sources D_l and D_r are effective when matching persons between sources D'_l and D'_r. The DL networks displayed in Figure 1 give an evaluation of the statistical relevance of each similarity function in a given application context.

[7]Nonetheless, blocking and few-shot learning are not mutually exclusive. The modular architecture displayed in Figure 1 can be easily combined with other pre-processing/blocking layers preceding the similarity encoding calculations that can further reduce the computational requirements.

[8]See Appendix B for the list of available pre-defined similarity functions with native implementations.

4.3 Computation and Modularity

Besides using the few-shot property to reduce memory requirements, our approach follows the previous literature and does not store the complete cross-product of the left and right data sets in memory. Instead, we use a data generator which iterates over the combinations of left and right data sets and feeds input pairs to the matching system using indexing methods (see also Doll et al., 2021).

Concerning the computation time, the properties of similarity encoders and the architecture displayed in Figure 1 allow our implementation to efficiently scale on modern computation clusters with multiple processing units. The scalability of the architecture stems from two points. First, the calculations of the similarity encoder are embarrassingly parallelizable. All similarity calculations perform only read operations on the input data and write operations in different memory locations of the output data. In terms of the visualization of Figure 2, a similarity encoder only reads the data from the cells of the left rectangular cuboid, while the output of each individual similarity calculation is written to one and only one cell of the right cuboid. These characteristics minimize the need for synchronization among multiple threads and processes to perform the encoding calculations in parallel and allow the computations to be spread across multiple processing units and computing nodes. Second, the calculations of matching predictions for the field pairs of D_l' and D_r' are independent of each other that makes the architecture modular across fields. Each field network uses only a slice of the input data with information on a single field pair to arrive at its matching prediction. Thus, the intermediate field's matching predictions enclosed in the annotated rectangle P of Figure 1 can be performed on different compute nodes and the results can be gathered at completion on the main node responsible for the record matching predictions.

5. Performance Evaluation

5.1 Benchmarking

In this section, we compare our EM framework to alternative open-source state-of-the-art EM systems using public benchmark datasets (Köpcke et al., 2010).[9] For the remainder of this section, we refer to our framework as MLMatch.

Hardware: We perform all calculations using a multi-socket system with AMD EPYC 7763 64-Core CPUs and 2 threats per core. The CPU speed is up to 2234.7 MHz and the system's total available memory amounts to 1.0Ti of RAM. The system uses two NVIDIA A100-PCIE GPUs with 40960 MiB each.

[9]The datasets can be accessed with this link.

Datasets: We test MLMatch in three different benchmarking matching tasks. The associated data are publicly available and comprise three datasets for each matching task: a left dataset, a right dataset and a two-column table representing the correct matches. As illustrated in Table 1, for two of the benchmark tasks (*Abt-Buy* & *Amazon-GoogleProducts*) the challenge is to correctly match (product-level) e-commerce data. In the third task (*DBLP-ACM*), the challenge is to link the bibliographic information from various computer science conferences.

The matching tasks further refer to *structured* and, in some cases, *dirty* data with the number of fields varying between three and four. Moreover, some fields do not have missing values (see the first number in the parentheses), while others have some missing values (indicated by the parentheses' second digit). The matching tasks cover *m:1* and *1:1* relationships between records from the *left* and *right* datasets (column *Rel.* in Table 1), which consist of up to 3.2 thousand entries.[10]

Table 1: Benchmark Data Summary Statistics

		Left		Right				
Dataset	Domain	Fields	#Records	Fields	#Records	#Matches	Rel.	Dirty
DBLP-ACM	Bibliographic	4 (4+0)	2,614	4 (4+0)	2,294	2,224	1:1	
Abt-Buy	E-commerce	3 (2+1)	1,081	4 (2+2)	1,092	1,097	m:1	✓
Amazon-GoogleProducts	E-commerce.	4 (2+2)	1,363	4 (3+1)	3,226	1,300	m:1	✓

Note: All datasets are provided by Köpcke et al. (2010). The first (second) digit in the parentheses following the field count indicates the number of fields without (with) missing information. The column *Dirty* indicates whether values of one field are (incorrectly) stored in other fields (Mudgal et al., 2018). For instance, in a subset of the *Buy* dataset, a product's *description* includes the *manufacturer*'s name although there is a separate field for this set of information.

Configuration: When calculating MLMatch's benchmark performance statistics, we use a largely uniform model calibration across all matching tasks. To enable comparability, we thus set the batch size to 90 and assign the `learning_rate` hyperparameter to 0.0001. Moreover, we always rely on the default network architecture throughout all set-ups. Differences, on the other hand, exist concerning the number of training epochs (1000 for *DBLP-ACM* and 2000 for *Abt-Buy* & *Amazon-GoogleProducts*), and the training ratio (0.15 for *DBLP-ACM*, 0.3 for *Amazon-GoogleProducts* and *Abt-Buy*). Additionally, we use a custom discrete similarity function to handle missing values. The custom function is used alongside the pre-defined ones, which higlights the flexibility of MLMatch's usage.[11] The custom function returns a value of one if the two input values are non-missing and equal; and zero otherwise. The definition of a meaningful custom similarity function does not require deep knowledge on the data corpora. Instead, rough eyeballing suffices

[10]In case of an *m:1* relationship, there might exist multiple matches from the *right* dataset for one observation of *left*, and vice versa.

[11]See Appendix 3 - 5 for the specification of the similarity map in each matching task.

in most cases, and – importantly – the inclusion of custom functions only has a small effect on the F1-scores in the three EM tasks.

Evaluation: To ensure the reliability of MLMatch's stated performance, we apply K-fold cross-validation with $K = 5$. The reported results correspond to the mean value of the five iterations. Table 2 illustrates the MLMatch's F1-scores in comparison to the current state-of-the-art EM system. Concerning the *DBLP-ACM* matching task, MLMatch performs well as opposed to other EM systems. It delivers the second highest F1-score of 99.1 which is only 0.1 points below the best-performing system. With regards to the more complicated matching tasks, MLMatch's relative performance is convincing. Our system outperforms its competitors in the *Abt-Buy* matching task with an F1-score that is 12.9 points higher compared to CorDEL, the second-best-performing system. However, the interpretation of this result should be approached with caution due to the availability of only five benchmark cases. Moreover, when predicting matching entities in the *Amazon-GoogleProducts* case, MLMatch's performance is superior to all but one systems with a F1-score of 84.9.

In summary, MLMatch performs well on all the benchmark datasets we use. In most cases, it outperforms alternative state-of-the-art EM systems.

Table 2: Benchmark Performance

		F1-score		
EM System	Source	DBLP - ACM	Abt - Buy	Amazon - GoogleProducts
Magellan	Mudgal et al. (2018)	98.4	43.6	49.1
DeepER	Ebraheem et al. (2018)	96.0	-	98.6
DeepMatcher	Mudgal et al. (2018)	98.4	62.8	69.3
Ditto	P. Li et al. (2021)	99.0	-	75.6
AdaMEL-hyb	Jin et al. (2021)	98.9	-	65.1
RuleSynth	Singh et al. (2017)	92.6	-	63.8
CorDEL	Z. Wang et al. (2020)	99.2	64.9	70.2
AutoFJ	P. Li et al. (2021)	97.7	61.3	-
ZeroER	Wu et al. (2020)	96.0	52.0	48.0
MLMatch		99.1	77.8	84.9
MLMatch **Rank**		2.	1.	2.

Note: This table illustrates the performance of current state-of-the-art EM systems in three different cases benchmarked against MLMatch. The illustrated performance statistics refers to the F-Score, defined as $2 \times \frac{precision \times recall}{precision + recall}$. The table further provides information on the system's authors. As the original paper that introduces Magellan (Konda et al., 2016) includes no performance statistics, we use information provided by Mudgal et al. (2018). Moreover, the indicated DeepMatcher statistics refer to the one of the four available DL models (SIF, RNN, Attention, and Hybrid) yielding the highest F1-scores as shown by Mudgal et al. (2018). We use the strategy for Z. Wang et al. (2020) who offer statistics for three different variations of CorDEL (Attention, Context Attention and and a simplified version named "Sum"). When reporting F1-scores of their unsupervised system ZeroER, Wu et al. (2020) only provide rounded values, explaining why the first decimal place is always zero. The "-" sign indicates that no performance statistics for the respective benchmark dataset are available.

5.2 A Use Case: OCR-Extracted Historical Firm-Level Data

In this section, we offer another use case. This use case refers to the matching of entities from repeated annual cross-sections of firm data extracted via OCR from historical German yearbooks. Gram et al. (2022) describe the data extraction process, detail the underlying relational data model, and provide summary statistics for a subset of the fields extracted for the period 1920 - 1932. The database is implemented according to the FAIR principles and will be made fully available the the public in the upcoming years.

We chose this set-up for the use case because (i) matching firms across sources is common for scientific studies in economics and finance, and (ii) it represents a challenging EM task. The task's challenges stem from multiple factors introducing variation between firms in the left (the cross-section in $t = t$) and the right datasets (the cross-section from $t = t + 1$): First, the OCR is imperfect. Hence, any variable extraction procedure that converts unstructured OCR-extracted text to structured formats leads to *dirty* data. Second, type-setting conventions were time-inconsistent. For example, the firm's address might be included in its name in one year; however, it might be left out in another year. A second example refers to inconsistent abbreviations. Third, the actual values of the fields might change between the two years. For example, a firm might change its name, move its headquarters to another location, or rephrase its purpose statement. In short, the set-up represents an appropriate testing and evaluation case for MLMatch.[12]

To create the training data, we first randomly pick 500 firm entries from the data source's 15th volume, corresponding to the financial year of 1910/11. Afterward, we manually identify these firms' representation in the 16th volume. After excluding (i) wrongly identified firms (i.e., an OCR text line that does not represent a firm that was incorrectly predicted to be one), and (ii) firms that have no representation in the 16th volume (e.g., due to bankruptcy), the labeled data consists of 367 entity pairs. The 10 selected fields for the EM task are relatively time-consistent across the volumes of the original source. These fields include, for instance, the firms' names as well as a range of date fields (e.g., founding and registration dates).[13] On the other hand, we do not account for fields that tend to change frequently over time, such as a firm's profit and loss statements or balance sheets.

We perform the ANN's training using the cross join of random samples representing 30% (110 entity pairs) of the labeled data. For string fields we feed six different similarity functions per field to the network (Levenshtein, Jaro-Winkler distances, and partial, token sort, token set, and partial token set ratios). Concerning date-related fields we rely on one (binary) discrete similarity function. This is because the representations of two different dates might have high string similarities even if they refer to different points in time (see

[12]See Adam et al. (2021); Cule et al. (2020); Karapanagiotis (2019); Poukens (2018) for a more detailed explanation of the requirements and pitfalls related to firm data extraction using OCR technologies.

[13]The keys in the `similarity_map` included in Listing 6 represent the labels of all 10 fields.

the discussion in Section 3.2).

We then perform the evaluation of the trained classifier on the the cross join of 70% (257 entity pairs) of the labeled data, as indicated by the Columns 3 and 4 in Table 3. When evaluating the matching performances delivered by MLMatch, we follow the K-fold cross-validation approach with $K = 5$. Thus, each row in Table 3 corresponds to one iteration (Column 1). In all of the five iterations, we set the `batch_size` to 90, training ratio to 0.3, and the `learning_rate` to 0.01. Moreover, in each of the iterations we train the models on 1,000 epochs.

As illustrated by Columns 5 to 12, MLMatch's performance is strong. The number of false positive predictions range between zero and four, and the count of false negative matches varies between zero and five across the five iterations. These numbers correspond to accuracy scores of no less than 99.9 and F-scores always above or equal to 99.

Table 3: Benchmark Performance

(1) Iteration	(2) Data Set	(3) #Test Matches	(4) Test Fraction	(5) TP	(6) FP	(7) TN	(8) FN	(9) Accuracy	(10) Precision	(11) Recall	(12) F-Score
1	Test	257	0.7	256	0	6430	2	100	100	99,2	99,6
2	Test	257	0.7	253	0	6430	5	99,9	100	98,1	99,0
3	Test	257	0.7	256	2	6428	2	99,9	99,2	99,2	99,2
4	Test	257	0.7	257	0	6430	1	100	100	99,6	99,8
5	Test	257	0.7	258	4	6426	0	99,9	98,5	100	99,2

Note: This table illustrates MLMatch's performance when matching repeated cross-sections of firm data extracted via OCR from historical German yearbooks. Column 1 indicates the K-fold cross-validation iteration. Column 3 shows the number of matching record pairs constituting the test sample's foundation. Column 4 shows the test fractions which equals $1 - training\_fraction$. Columns 5 to 8 show the number of true positive, false positive, true negative, and false negative predicted matching pairs, respectively. Columns 9 to 12 highlight the resulting performance metrics, including accuracy $(= \frac{TP+TN}{TP+TN+FP+FN})$, precision $(= \frac{TP}{TP+FP})$, recall $(= \frac{TP}{TP+FN})$, and the F-Score $(= 2 \times \frac{precision \times recall}{precision+recall})$.

6. Conclusion

We contribute to the literature on semi-automated EM frameworks that aim to reduce the human-in-the-loop requirements when linking data sources. In doing so, we combine and extent recent developments in economics, finance, and computer science EM literature.

In particular, we propose and implement a modular EM recommendation framework based on ANNs, the state-of-the-art approach in computer science, and similarity encoding, following the best practices from linking data in finance and economics. Our recommendation framework can be applied in various domains.

Compared to alternative state-of-the-art EM systems, our framework proves beneficial for five reasons. First, it reduces the requirements for human expertise on the content of the linked data sources. Second, using a similarity encoder enables training

ANNs without relying on word embedding, which tend to have greater technical expertise requirements. Third, not relying on word embeddings can, at the same time, reduce the computational complexity associated with training embedding-based ANNs. Fourth, due to the properties of similarity encoders, the encoding calculations can be easily parallelized, and our approach can be scaled up using multiple computing nodes. Fifth, the similarity encoder is not subject to the Out-of-Vocabulary issues that word-embedding-based EM systems face when used with dirty or multilingual data.

Morever, we use three benchmark cases to demonstrate that our approach is on-par and, in most cases, outperforms the majority of alternative state-of-the-art systems. When matching product-level e-commerce records in the *Abt-Buy* EM task, it exhibits an F1-score of 77.8, which is 12.9 points higher compared to the second-best-performing system. In the other two benchmark cases (*DBLP-ACM* and *Amazon-GoogleProduct*), our approach yields the second highest F1-scores (99.1 and 84.9, respectively) when compared to nine alternative systems.

Our similarity encoding matching framework likewise performs well in tasks that are more common for research in economics and finance. We use dirty firm-level data extracted via OCR from historical sources to illustrate our framwork's relevance in these fields. Our framework achieves an average F1-score of 99.36 when conducting 5-fold cross-validation in this set-up.

References

Abramitzky, R., Boustan, L., Eriksson, K., Feigenbaum, J., & Pérez, S. (2021, September). Automated linking of historical data. *Journal of Economic Literature*, *59*(3), 865–918. doi: 10.1257/jel.20201599

Abramitzky, R., Boustan, L., Jacome, E., & Perez, S. (2021, February). Intergenerational mobility of immigrants in the united states over two centuries. *American Economic Review*, *111*(2), 580–608. doi: 10.1257/aer.20191586

Abramitzky, R., Mill, R., & Pérez, S. (2020). Linking individuals across historical sources: A fully automated approach. *Historical Methods: A Journal of Quantitative and Interdisciplinary History*, *53*(2), 94–111. doi: 10.1080/01615440.2018.1543034

Adam, S., Annaert, J., Buelens, F., Coüasnon, B. B., Cule, B., de Vicq, A., ... Riva, A. (2021, December). Data extraction and matching The EurHisFirm experience. In *Methodological advances in the extraction and analysis of historical data.* Chicago/Virtual, United States. (https://hal.science/hal-03828381)

Antoni, M., & Schnell, R. (2019). The Past, Present and Future of the German Record Linkage Center (GRLC). *Jahrbücher für Nationalökonomie und Statistik*, *239*(2), 319–331. doi: 10.1515/jbnst-2017-1004

Bachmann, M. (2021, October). *Maxbachmann/RapidFuzz: Release 1.8.0.* Zenodo. doi: 10.5281/zenodo.5584996

Bailey, M. J., Cole, C., Henderson, M., & Massey, C. (2020). How well do automated linking methods perform? Lessons from US historical data. *Journal of Economic Literature*, *58*(4), 997–1044.

Bartram, S. M., Hou, K., & Kim, S. (2022). Real effects of climate policy: Financial constraints and spillovers. *Journal of Financial Economics*, *143*(2), 668–696.

Bilenko, M., & Mooney, R. J. (2003). Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 39–48).

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the association for computational linguistics*, *5*, 135–146.

Buggle, J., Mayer, T., Sakalli, S. O., & Thoenig, M. (2023, January). The refugee's dilemma: Evidence from jewish migration out of nazi germany*. *The Quarterly Journal of Economics*, *138*(2), 1273–1345. doi: 10.1093/qje/qjad001

Chollet, F., & others. (2015). *Keras.* https://keras.io.

Christen, P. (2012). *Data matching: Concepts and techniques for record linkage, entity resolution, and duplicate detection.* Springer Publishing Company, Incorporated.

Cule, B., buelens, F., poukens, J., annaert, J., & richer, J. (2020, December). *EurHisFirm M6.2: Data connecting case study.* Zenodo. doi: 10.5281/zenodo.4309048

Doll, H., Gabor-Toth, E., & Schild, C.-J. (2021, May). *Linking deutsche bundesbank company data.* Deutsche Bundesbank, Research Data and Service Centre.

Dunn, H. L. (1946). Record linkage. *American Journal of Public Health and the Nations Health*, *36*(12), 1412–1416.

Ebraheem, M., Thirumuruganathan, S., Joty, S., Ouzzani, M., & Tang, N. (2018). Distributed representations of tuples for entity resolution. *Proceedings of the VLDB Endowment*, *11*(11), 1454–1467.

Fan, W., Jia, X., Li, J., & Ma, S. (2009). Reasoning about record matching rules. *Proceedings of the VLDB Endowment*, *2*(1), 407–418.

Feigenbaum, J. J. (2016). *Automated census record linking: A machine learning approach* (Working Paper). Massachusetts: Harvard University. (https://scholar.harvard.edu/jfeigenbaum/publications/automated-census-record-linking)

Ferrie, J. P. (1996). A new sample of males linked from the public use microdata sample of the 1850 U.S. federal census of population to the 1860 U.S. federal census manuscript schedules. *Historical Methods: A Journal of Quantitative and Interdisciplinary History*, *29*(4), 141–156. doi: 10.1080/01615440.1996.10112735

Ferrie, J. P. (2005). History lessons: The end of American exceptionalism? Mobility in the United States since 1850. *Journal of Economic Perspectives*, *19*(3), 199–215.

Fisman, D., Grogin, J., Margalit, O., & Weiss, G. (2022). The normalized edit distance with uniform operation costs is a metric. In H. Bannai & J. Holub (Eds.), *33rd annual symposium on combinatorial pattern matching (CPM 2022)* (Vol. 223, pp. 17:1–17:17). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi: 10.4230/LIPIcs.CPM.2022.17

Florackis, C., Louca, C., Michaely, R., & Weber, M. (2023). Cybersecurity risk. *The Review of Financial Studies*, *36*(1), 351–407.

Ghosh, A., Hwang, S. I. M., & Squires, M. (2023). Links and legibility: Making sense of historical US Census automated linking methods. *Journal of Business & Economic Statistics*, *0*(0), 1–12.

Gokhale, C., Das, S., Doan, A., Naughton, J. F., Rampalli, N., Shavlik, J., & Zhu, X. (2014). Corleone: Hands-off crowdsourcing for entity matching. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data* (pp. 601–612).

González-Carrasco, I., Jiménez-Márquez, J. L., López-Cuadrado, J. L., & Ruiz-Mezcua, B. (2019, June). Automatic detection of relationships between banking operations using machine learning. *Information Sciences*, *485*, 319–346. doi: 10.1016/j.ins.2019.02.030

Gram, D., Karapanagiotis, P., Liebald, M., & Walz, U. (2022). Design and implementation of a historical german firm-level financial database. *ACM Journal of Data and Information Quality (JDIQ)*, *14*(3), 1–22.

Hamming, R. W. (1950). Error detecting and error correcting codes. *The Bell system technical journal*, *29*(2), 147–160.

Helgertz, J., Price, J., Wellington, J., Thompson, K. J., Ruggles, S., & Fitch, C. A. (2022). A new strategy for linking U.S. historical censuses: A case study for the IPUMS multigenerational longitudinal panel. *Historical Methods: A Journal of Quantitative and Interdisciplinary History*, *55*(1), 12–29. doi: 10.1080/01615440.2021.1985027

Jin, D., Sisman, B., Wei, H., Dong, X. L., & Koutra, D. (2021). Deep transfer learning for multi-source entity linkage via domain adaptation. In *Proceedings of the VLDB endowment* (Vol. 15, pp. 465–477). doi: 10.14778/3494124.3494131

Kamlah, J., Schmidt, T., & Shigapov, R. (2022). *Extracting research data from historical documents with eScriptorium and Python.* doi: 10.5281/zenodo.7373134

Karapanagiotis, P. (2019, September). *EurHisFirm D5.1: Technical document on national data models* (Tech. Rep.). Zenodo. doi: 10.5281/zenodo.3467926

Kasai, J., Qian, K., Gurajada, S., Li, Y., & Popa, L. (2019). Low-resource deep entity resolution with transfer and active learning. In *Proceedings of the 57th annual meeting of the association for computational linguistics* (pp. 5851–5861).

Konda, P., Das, S., Doan, A., Ardalan, A., Ballard, J. R., Li, H., ... others (2016). Magellan: Toward building entity matching management systems over data science stacks. *Proceedings of the VLDB Endowment*, *9*(13), 1581–1584.

Köpcke, H., Thor, A., & Rahm, E. (2010). Evaluation of entity resolution approaches on real-world match problems. *Proceedings of the VLDB Endowment*, *3*(1-2), 484–493.

Levenshtein, V. I. (1965). Binary codes capable of correcting deletions, insertions, and reversals. In *Proceedings of the USSR academy of sciences* (Vol. 163, pp. 845–848). Russian Academy of Sciences.

Li, P., Cheng, X., Chu, X., He, Y., & Chaudhuri, S. (2021). Auto-FuzzyJoin: Auto-program fuzzy similarity joins without labeled examples. In *Proceedings of the 2021 international conference on management of data* (pp. 1064–1076).

Li, Y., Li, J., Suhara, Y., Doan, A., & Tan, W.-C. (2020). Deep entity matching with pre-trained language models. *Proceedings of the VLDB Endowment*, *14*(1), 50–60.

López-Cuadrado, J. L., González-Carrasco, I., Leonardo López-Hernández, J., Martínez-Fernández, P., & Martínez-Fernández, J. L. (2020). Automatic learning framework for pharmaceutical record matching. *IEEE access : practical innovations, open solutions*, *8*, 171754–171770. doi: 10.1109/ACCESS.2020.3024558

Mannheim, U. (2019). *Handbuch der deutschen aktiengesellschaften.* urn:nbn:de:bsz:180-dighop-181. Heppenheim (Bergstr.), Berlin: Hoppenstedt.

Marzal, A., & Vidal, E. (1993). Computation of normalized edit distance and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *15*(9), 926–932. doi: 10.1109/34.232078

Meduri, V. V., Popa, L., Sen, P., & Sarwat, M. (2020). A comprehensive benchmark framework for active learning methods in entity matching. In *Proceedings of the 2020 ACM SIGMOD international conference on management of data* (pp. 1133–1147).

Mudgal, S., Li, H., Rekatsinas, T., Doan, A., Park, Y., Krishnan, G., … Raghavendra, V. (2018). Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 international conference on management of data* (pp. 19–34).

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532–1543).

Persson, P. (2020). Social insurance and the marriage market. *Journal of Political Economy*, *128*(1), 252–300. doi: 10.1086/704073

Piktus, A., Edizel, N. B., Bojanowski, P., Grave, E., Ferreira, R., & Silvestri, F. (2019, June). Misspelling oblivious word embeddings. In *Proceedings of the 2019 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 1 (long and short papers)* (pp. 3226–3234). Minneapolis, Minnesota: Association for Computational Linguistics. (https://aclanthology.org/N19-1326) doi: 10.18653/v1/N19-1326

Poukens, J. (2018, December). *EurHisFirm D4.2: Report on the inventory of data and sources* (Tech. Rep.). Zenodo. doi: 10.5281/zenodo.3246457

Price, J., Buckles, K., Van Leeuwen, J., & Riley, I. (2021). Combining family history and machine learning to link historical records: The Census Tree data set. *Explorations in Economic History*, *80*, 101391.

Qian, K., Popa, L., & Sen, P. (2017). Active learning for large-scale entity resolution. In *Proceedings of the 2017 ACM on conference on information and knowledge management* (pp. 1379–1388).

Rodriguez-Lujan, I., & Huerta, R. (2016). An algorithm for matching heterogeneous financial databases: A case study for COMPUSTAT/CRSP and I/B/E/S databases. *Applied Economics and Finance*, *3*(1), 161–172. doi: 10.11114/aef.v3i1.1164

Rozinek, O., & Mareš, J. (2021). The duality of similarity and metric spaces. *Applied Sciences*, *11*(4), 1–18. doi: 10.3390/app11041910

Singh, R., Meduri, V. V., Elmagarmid, A., Madden, S., Papotti, P., Quiané-Ruiz, J.-A., … Tang, N. (2017). Synthesizing entity matching rules by examples. *Proceedings of the VLDB Endowment*, *11*(2), 189–202.

Stal, C., De Sloover, L., Verbeurgt, J., & De Wulf, A. (2022). On finding a projected coordinate reference system. *Geographies*, *2*(2), 245–257.

Wang, J., Kraska, T., Franklin, M. J., & Feng, J. (2012). CrowdER: Crowdsourcing entity resolution. *Proceedings of the VLDB Endowment*, *5*(11), 1483–1494.

Wang, J., Li, G., Yu, J. X., & Feng, J. (2011). Entity matching: How similar is similar. *Proceedings of the VLDB Endowment*, *4*(10), 622–633.

Wang, Z., Sisman, B., Wei, H., Dong, X. L., & Ji, S. (2020). CorDEL: A contrastive deep learning approach for entity linkage. In *2020 IEEE international conference on data mining (ICDM)* (pp. 1322–1327). IEEE.

Wojcik, S., Bijral, A. S., Johnston, R., Lavista Ferres, J. M., King, G., Kennedy, R., . . . Lazer, D. (2021, January). Survey data and human computation for improved flu tracking. *Nature communications*, *12*(1), 194. doi: 10.1038/s41467-020-20206-z

Wu, R., Chaba, S., Sawlani, S., Chu, X., & Thirumuruganathan, S. (2020). Zeroer: Entity resolution using zero labeled examples. In *Proceedings of the 2020 ACM SIGMOD international conference on management of data* (pp. 1149–1164).

Zhao, C., & He, Y. (2019). Auto-em: End-to-end fuzzy entity-matching using pre-trained deep models and transfer learning. In *The world wide web conference* (pp. 2413–2424).

A. Proofs of section 3.2

A.1 Proof of proposition 1

Suppose that X is metrizable and let d be a metric on X. Without loss of generality, we can assume that d takes values in $[0, 1]$. If this is not the case and d is not bounded, we can define $\tilde{d}(x, y) = d(x, y)/ (1 + d(x, y))$. The function $\tilde{d}$ is a metric on X and has values in $[0, 1]$. If, instead, d is bounded but takes values greater than one, we can define $\tilde{d}(x, y) = d(x, y)/ \sup_{x',y'\in X} d(x', y')$ and use $\tilde{d}$ instead of d. Define $s(x, y) = 1-d(x, y)$ for all $x, y \in X$. The function s is (i) symmetric and satisfies (ii) $s(x, y) = 1 \iff d(x, y) = 0 \iff x = y$ and (iii)

$$s(x, y) + 1 = 2 - d(x, y) \geq 1 - d(x, z) + 1 - d(z, y) = s(x, z) + s(z, y).$$

For the reverse direction, suppose that s is a normalized similarity on X. Define $d(x, y) = 1 - s(x, y)$ for all $x, y \in X$. The function d is (i) symmetric, and satisfies (ii) $d(x, y) = 0 \iff s(x, y) = 1 \iff x = y$ and (iii)

$$d(x, y) = 1 - s(x, y) \leq 1 - s(x, z) + 1 - s(z, y) = d(x, z) + d(z, y).$$

A.2 Proof of proposition 2

The similarity map is not an operator because its domain $\mathcal{N}$ is not a vector space, since for any real number $\lambda > 1$ and any normalized similarity function s, λs is not a normalized similarity function. To prove the proposition's claims, we define an operator acting on sets of bounded functions as the similarity map acts on sets of similarity functions.

We use the following notation to facilitate the exposition. For a set A, and a normed space B, we use $\mathcal{B}(A, B)$ to denote the set of bounded functions from A to B. If $B = \mathbb{R}$, we simply write $\mathcal{B}(A)$. For any natural number n, we use the notation A^n to denote the n-Cartesian product $A \times A \times \ldots \times A$.

Consider the vector space

$$\mathcal{X} = \mathcal{B}\left(X_1^2\right)^{m_1} \times \mathcal{B}\left(X_2^2\right)^{m_2} \times \ldots \times \mathcal{B}\left(X_k^2\right)^{m_k},$$

endowed with the product norm

$$\|\left(f_1^1, \ldots, f_{m_1}^1, \ldots, f_1^2, \ldots, f_{m_2}^2, \ldots, f_1^k, \ldots, f_{m_k}^k\right)\|_{\mathcal{X}} = \sum_{i=1}^{k} \sum_{j=1}^{m_i} \|f_j^i\|_\infty,$$

and the vector space $\mathcal{Y} = \mathcal{B}\left((X_1 \times \ldots \times X_k)^2, \mathbb{R}^m\right)$, where $m = \sum_{i=1}^k m_i$ and $\mathbb{R}^m$ is endowed with the ℓ^1 norm. Consider the operator $\hat{\mathrm{S}}\colon \mathcal{X} \to \mathcal{Y}$, such that for every $f = \left(f_1^1, \ldots, f_{m_1}^1, \ldots, \ldots, \ldots, f_1^k, \ldots, f_{m_k}^k\right) \in \mathcal{X}$, $\hat{\mathrm{S}}f$ is equal to a function c_f defined by

$$c_f\colon \left(x^l, x^r\right) \mapsto \left(f_1^1(x_1^l, x_1^r), \ldots, f_{m_1}^1(x_1^l, x_1^r), \ldots, f_1^k(x_k^l, f_k^r), \ldots, f_{m_k}^k(x_k^l, x_k^r))\right).$$

The restriction of the operator $\hat{\mathrm{S}}$ on $\mathcal{N}$ gives the similarity map S defined by eq. (3).

By its definition, $\hat{\mathrm{S}}$ is linear and for any $f \in \mathcal{X}$, we have

$$\|\hat{\mathrm{S}}f\|_1 = \|c_f\|_1 = \|c_{\left(f_1^1,\ldots,f_{m_1}^1,\ldots,f_1^k,\ldots,f_{m_k}^k\right)}\|_1 = \sum_{i=1}^{k} \sum_{j=1}^{m_i} \|f_j^i\|_\infty = \|f\|_{\mathcal{X}},$$

which implies that $\hat{\mathrm{S}}$ (i) is an isometry, (ii) is bounded, (iii) is continuous, and (iv) has operator norm

$$\|\hat{\mathrm{S}}\| = \sup_{\|f\|_{\mathcal{X}}=1} \|\hat{\mathrm{S}}f\|_1 = 1.$$

This also shows that the similarity map is an isometry.

Lastly, suppose that $\mathcal{I}_1$ and $\mathcal{I}_2$ are two sets of instructions in $\mathcal{N}$ and $\alpha \in [0,1]$. Convex combinations of normalized similarities are normalized similarities, thus $\mathcal{I}_\alpha = \alpha\mathcal{I}_1 + (1-\alpha)\mathcal{I}_2$ is in $\mathcal{N}$. By the linearity of $\hat{\mathrm{S}}$, we get

$$\mathrm{S}\mathcal{I}_\alpha = \hat{\mathrm{S}}\mathcal{I}_\alpha = \alpha\mathrm{S}\mathcal{I}_1 + (1-\alpha)\mathrm{S}\mathcal{I}_2 = \alpha c_{\mathcal{I}_1} + (1-\alpha)c_{\mathcal{I}_2},$$

which shows that convex combinations of instructions are mapped to convex combinations of encoders.

B. Implementation Details

The implementation of the similarity encoder is written in `C++`. Two bindings exist, making the encoder's functionality accessible in both `Python` (`pymlmatch`) and `R` (`rmlmatch`). Both bindings use the Keras library (Chollet & others, 2015) for the EM ANNs. The similarity encoder and the two bindings are distributed under the Expat license. Listings 1 and 2 give workflow examples for training of a matching model in both the `Python` and `R` libraries.

Listing 1: `Python` Binding Syntax

```python
model = match.MatchingModel(similarity_map)

model.compile(
    loss = "binary_crossentropy",
    optimizer = tensorflow.keras.optimizers.Adam(
        learning_rate = 1e-3),
    metrics = [
        tensorflow.keras.metrics.TruePositives(),
        tensorflow.keras.metrics.FalsePositives(),
        tensorflow.keras.metrics.TrueNegatives(),
        tensorflow.keras.metrics.FalseNegatives()])

model.fit(
    left_train,
    right_train,
    matches_train,
    epochs = 2000,
    batch_size = 312,
    mismatch_share = 0.2,
    shuffle = True)

train_eval = model.evaluate(
    left_test,  right_test, matches_test)

predictions = model.predict(
    left, left)

suggestions = model.suggest(
    left, left, count = 3)
```

Listing 2: R Binding Syntax

```r
model <- matching_model(similarity_map)

model |> compile(
  loss = keras::loss_binary_crossentropy(),
  optimizer = keras::optimizer_adam(
    learning_rate = 1e-3),
  metrics = list(
    keras::metric_true_positives(),
    keras::metric_false_positives(),
    keras::metric_true_negatives(),
    keras::metric_false_negatives()))

model |> fit(
    left_train,
    right_train,
    matches_train,
    epochs = 2000L,
    batch_size = 312L,
    mismatch_share = 0.2,
    shuffle = TRUE)

model |>
  evaluate(left_test, right_test, matches_test)

predictions <- model |>
  predict(left, right)

suggestions <- model |>
  suggest(left, right, count = 3)
```

Initialization and Compilation: The initialization of a matching model requires a similarity map that defines

1. the fields supposed to be used from the `left` and `right` data sets,
2. the way they relate to one another, and
3. which similarity functions are to be considered (i.e., how to encode the data).

Both the `Python` and `R` libraries offer a set of commonly used pre-defined similarity and ratio functions with native implementations. These are the `discrete`, `euclidean`, `gaussian`, `levenshtein`, `jaro`, and `jaro_winkel` distances, and the `partial`, `token_sort`, `partial_token_sort`, `token_set`, and `partial_token_set` ratios. The `discrete` similarity can be used with fields of any type, while the `gaussian` and `euclidean` distances work only with numerical data. The remaining functions can be applied to string fields. The string similarity and ratio calculations use the RapidFuzz `C++` library (Bachmann, 2021). Aside from the pre-defined functions, both `pymlmatch` and `rmlmatch` allow the caller to use custom similarity functions.

Multiple similarity functions can be used for each association of fields. Moreover, in the case where the associated fields from the `left` and `right` data sets do not share a common name, they can be added to the similarity map using the ∼ operator (i.e., `left_name∼right_name`). If the associated fields have the same name, it suffices to only pass the name once.[14]

A `matching_model` object spans by default an ANN with n *Field Networks* and one subsequent, overarching *Record Network*, where n equals the number of fields included in the similarity map (see Figure 1 in the main body of the text for a visualization of the architecture). If not specified differently, each *Field Network* has four hidden dense layers. The unit counts of these dense layers equal the maximum of i) the number of similarity functions used for the respective field multiplied by $15/d$, where d is the depth position of the layer. Thus, assuming the number of similarity functions is sufficiently large, the unit counts reduce toward the end of the *Field Networks*. The layers use a rectified linear unit (relu) activation function. The *Field Networks*'s output layer is a single unit classifying dense layer. The *Record Network* is built using similar logic and has the same similar values for the network's width and depth.

The `matching_model` class is derived from the Keras library's model classes (Chollet & others, 2015). Objects of the `matching_model` class need to be compiled before their use. The `matching_model` class's `compile` method wraps the Keras model class's `compile` method. The options and the keyword arguments of Keras's `compile` can be used with `matching_model`'s `compile`.

Fit and Evaluation: Fitting a `matching_model` object is performed using the `fit` function. The `fit` function extends the Keras's functionality to adapt it to EM requirements. Model training requires three data sets: (i) & (ii) the left and right data with the training records from the data sources to be linked, and (iii) the `matches` data, which is a two-column data set with the indices of matching records of training data. Additional keyword arguments such as the number of training epochs or the learning rate can be set using the standard Keras keyword arguments. The mismatch training examples are automatically calculated by pairing the indices of `left` and `right` that are not recorded in the `matches` data. Because the total number of candidate entity pairs grows quadratically while the number of actual matching pairs only grows linearly with the number of observations `left` and `right` (Christen, 2012), the inclusion of all mismatches can lead to unbalanced training data. This inclusion can potentially lead to biased model fits with good accuracy but bad precision, recall, and F1-score metrics. The keyword argument `mismatch_share` can be used to restrict the number of used mismatches during training.

The performance of a `matching_model` fit can be assessed via the `evaluate` function which overrides the corresponding Keras function. Calling `evaluate` requires `left`, `right`, and `matches` evaluation data. The model is evaluated over all the potential entity

[14]Listings 3 - 5 provide examples of `similarity_map` specifications. The illustrated examples correspond to the benchmark performance evaluation tasks in Section 5.

pairs from the cross join between `left` and `right`, even if the `mismatch_share` was set to less than one during training.

Predictions and Suggestions: The fitted model can generate matching predictions in two separate ways. First, via the `predict` function which returns a vector of matching probabilities for all candidate record pairs from the cross join of `left` and `right`. Second, via the `suggest` function, which for each record in `left` returns n matching recommendations from `right` that exhibit the highest matching probability. The number of returned recommendations can be modified using the `count` keyword argument.

C. Listings

Listing 3: The Similarity Map Used for the *DBLP-ACM* Matching Task

```
similarity_map = {
    "title": ["levenshtein", "jaro_winkler", "partial", "token_sort", "token_set", "partial_token_set"],
    "authors": ["levenshtein", "jaro_winkler", "partial", "token_sort", "token_set", "partial_token_set"],
    "venue": ["levenshtein", "jaro_winkler", "partial", discrete_capital_letters],
    "year" : ["euclidean", "gaussian", discrete]}
```

Listing 4: The Similarity Map Used for the *Abt-Buy* Matching Task

```
similarity_map = {
    "description": ["levenshtein", "jaro_winkler", "partial", "token_sort", "token_set", "partial_token_set"],
    "name": ["levenshtein", "jaro_winkler", "partial", "token_sort", "token_set", "partial_token_set", discrete],
    "description~name": ["levenshtein", "jaro_winkler", "partial", "token_sort", "token_set", "partial_token_set"],
    "name~description": ["levenshtein", "jaro_winkler", "partial", "token_sort", "token_set", "partial_token_set"],
    "price" : ["levenshtein", "jaro_winkler", "partial", "token_sort", "token_set", "partial_token_set", discrete],
    "name~manufacturer" :: ["partial", "partial_token_set"],
    "description~manufacturer" :: ["partial", "partial_token_set"]}
```

Listing 5: The Similarity Map Used for the *Amazon-GoogleProducts* Matching Task

```
similarity_map = {
    "description": ["levenshtein", "jaro_winkler", "partial", "token_sort", "token_set", "partial_token_set"],
    "title~name": ["levenshtein", "jaro_winkler", "partial", "token_sort", "token_set", "partial_token_set"],
    "description~name": ["levenshtein", "jaro_winkler", "partial", "token_sort", "token_set", "partial_token_set"],
    "title~description": ["levenshtein", "jaro_winkler", "partial", "token_sort", "token_set", "partial_token_set"],
    "manufacturer": [
        "levenshtein", "jaro_winkler", "partial", "token_sort", "token_set", "partial_token_set", discrete],
    "price" : ["levenshtein", "jaro_winkler", "partial", "token_sort", "token_set", "partial_token_set",discrete]}
```

Listing 6: The Similarity Map Used When Matching Historical Firms

```
similarity_map = {
    "company_name": ["levenshtein", "jaro_winkler", "partial", "token_sort", "token_set", "partial_token_set"],
    "company_info_1" : ["levenshtein", "jaro_winkler", "partial", "token_sort", "token_set", "partial_token_set"],
    "company_info_2" : ["levenshtein", "jaro_winkler", "partial", "token_sort", "token_set", "partial_token_set"],
    "found_date": [discrete],
    "found_year": [discrete],
    "register_date": [discrete],
    "register_year": [discrete],
    "concession_date": [discrete],
    "concession_year": [discrete],
    "statue_change_date": [discrete],
    "company_name~company_info_1": [
        "levenshtein", "jaro_winkler", "partial", "token_sort", "token_set", "partial_token_set"],
    "company_name~company_info_2": [
        "levenshtein", "jaro_winkler", "partial", "token_sort", "token_set", "partial_token_set"],
    "company_info_1~company_info_2": [
        "levenshtein", "jaro_winkler", "partial", "token_sort", "token_set", "partial_token_set"]}
```

D. EM Approaches Used in Recent Finance and Economics Articles

Table 4: EM in Recent High-Ranked Finance and Economics Publications

(1) Journal	(2) Author(s)	(3) EM Task	(4) Entity Type	(5) Method
Journal of Finance	Griffin et al. (2023)	✓	Firms	Rule-based
Journal of Finance	Gathergood et al. (2023)	✓	Financial Products	Common ID
Journal of Finance	Sautner et al. (2023)	✓	Firms	Rule-based
Journal of Finance	Gormsen and Lazarus (2023)	✓	Firms	Common ID
Journal of Finance	Hsu et al. (2023)	✓	Firms	Rule-based
Journal of Finance	Cordell et al. (2023)	✓	Financial Products	Common ID
Quarterly Journal of Economics	Moscona and Sastry (2023)	✓	Crops	Rule-based
Quarterly Journal of Economics	Babina et al. (2023)	✓	Persons	Rule-based
Quarterly Journal of Economics	Ganong and Noel (2022)	✓	Financial Accounts	-
Quarterly Journal of Economics	Bai et al. (2022)	✓	Persons	Manual
Quarterly Journal of Economics	Buggle et al. (2023)	✓	Persons	Probabilistic

Note: This table lists the scientific articles from two issues of *The Journal of Finance* (Vol. 78, Issue 3) and *The Quarterly Journal of Economics* that use entity matching (EM). In total, the two reviewed issues have 24 articles. Of those, 11, or 46%, use EM. If different EM tasks are performed in an article, we list the method and entity type that correspond to the task that appears to be the technically most challenging from a data management perspective. Column 5 illustrates the applied methods. These are human computation-based manual EM, EM using an existing common identifier, and rule-based approaches relying on pre-defined matching thresholds for one string similarity function. Only Buggle et al. (2023) apply a technical more complex (probabilistic) matching approach, that likewise accounts for one similarity function. With regards to Ganong and Noel (2022), the authors provide no information on the applied EM approach, indicated by "-".

Recent Issues