Scoping: Towards Streamlined Entity Collections for Multi-Sourced Entity Resolution with Self-Supervised Agents

Leonard Traeger[1,2], Andreas Behrend[2] and George Karabatis[1][a]

[1]*Department of Information Systems, University of Maryland, Baltimore County, U.S.A.*

[2]*Institute of Computer and Communication Technology (ICCT), Technical University of Cologne, Germany*

Keywords: Data Cleaning and Integration, Entity Resolution, Entity Linkage, Data Quality, Deep Learning.

Abstract: Linking multiple entities to a real-world object is a time-consuming and error-prone task. Entity Resolution (ER) includes techniques for vectorizing entities (signature), grouping similar entities into partitions (blocking), and matching entity pairs based on specified similarity thresholds (filtering). This paper introduces scoping as a new and integral phase in multi-sourced ER with potentially increased heterogeneity and more unlinkable entities. Scoping reduces the space of candidate entity pairs by ranking, detecting, and removing unlinkable entities through outlier algorithms and reusable self-supervised autoencoders, leaving intact the set of true linkages. Evaluations on multi-sourced schemas show that autoencoders perform best in schemas relevant to each other, where they reduce entity collections to 77% and still contain all linkages.

1 INTRODUCTION

Entity Linkage (EL) is a core discipline in Entity Resolution (ER) and data management especially when dealing with integration tasks. The overall goal is to clarify a global entity profile such as a customer's address connected to all underlying data sources. *It is evident that linking entities between more than two data sources results in a significantly higher degree of heterogeneity and variance in data quality.* Rahm et al. define this issue as "multi-sourced" ER in which arbitrary numbers of sources and respective entity profiles refer to the same real-world object (Lerm et al., 2021). An entity profile e_i may be an attribute of a relational database schema "product id" or API service "product code" representing the real-world entity r "product identifier". The set of all entity profiles within one data source denotes as an entity collection $\mathcal{E}_i$, e.g., $\mathcal{E}_1 =$ "**O**rder_**C**ustomer-Oracle". Traditional EL solutions pass the raw entity collections of all entity profiles from one module to another, resulting in linkages that may not always reflect reality. In addition, there is a lot of computational power to be used for such inaccurate linkages (Papadakis et al., 2022). In order to solve this problem, we introduce scoping, a new phase in the EL pipeline that ranks entities and generates a subset $\mathcal{E}'$ of the raw entity collections $\mathcal{E}$ while leaving intact the major set of true linkages.

[a] https://orcid.org/0000-0002-2208-0801

Motivating Example: Figure 1 depicts an excerpt of a multi-sourced Entity Linkage problem between three entity collections sampled from three schemas of database vendors Oracle (E_1), MySQL (E_2), and SAP HANA (E_3), storing data about customers and products. Each entity collection contains different entity profiles representing relational attributes e_i. The brute-force approach of comparing each entity from one collection with all entities from the other collections results in 180 comparisons. We need a solution to this problem that reduces the number of comparisons by identifying a subset of the entity collections containing fewer unlinkable entities while keeping those with true linkages. Therefore, we introduce scoping, a new technique that reduces the space of potential linkages in multi-sourced ER. **Scoping** is performed between the signature and blocking phases of the EL pipeline, and it consists of ranking, sorting, and filtering the data quality of entity signatures. It is an integral phase of multi-sourced Entity Resolution that can yield streamlined entity collections. Following the Big Data Vs, we assume that the number of entity collections rises in "Volume" and comes along with "Veracity". Therefore, the more entity collections need to be linked between them, the more entities without corresponding linkage will also remain at

Traeger, L., Behrend, A. and Karabatis, G.
Scoping: Towards Streamlined Entity Collections for Multi-Sourced Entity Resolution with Self-Supervised Agents.
DOI: 10.5220/0012607500003690

In *Proceedings of the 26th International Conference on Enterprise Information Systems (ICEIS 2024) - Volume 1*, pages 107-115
ISBN: 978-989-758-692-7; ISSN: 2184-4992

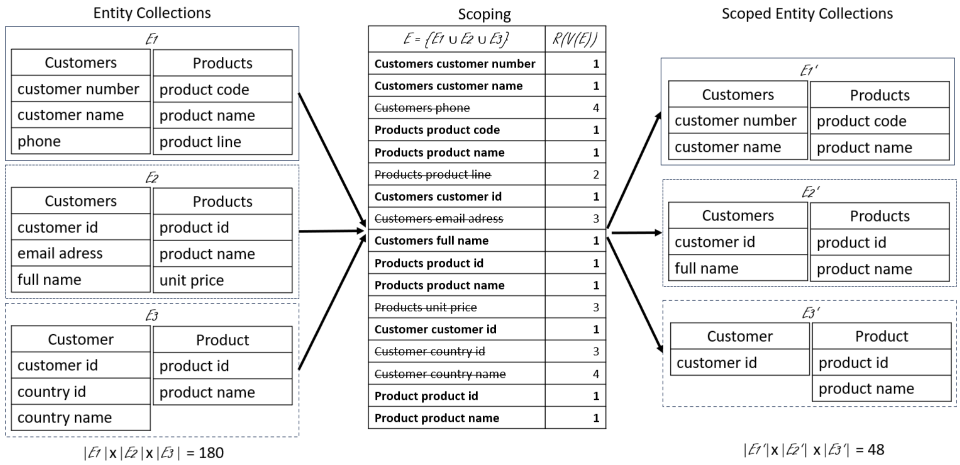

Figure 1: Scoping of multi-sourced Entity Collections (Example).

the end of the EL process. For example, as the entity collections become larger due to the rise in Volume, we observe an increase in heterogeneity and a decrease in Veracity. Figure 2 illustrates the increasing heterogeneity represented by the colored entity collections represented as ovals with unlinkable entities, while the linked ones are shown in the white overlapping oval. Scoping seeks entities with linkages represented in the overlapping oval. The contributions of this paper are:

- We introduce scoping, a novel and integral phase of Entity Linkage, positioned right after the signature and before the blocking phase (Figure 3), improving the quality of the search space of pair candidates (Section 3).
- We create entity ranking methods for scoping by relating them to anomaly detection and adapt Z-score, LOF, PCA, autoencoders, and ensembles as entity ranking methods (Section 3.2).
- We evaluate our scoping approach based on additional performance metrics on a new real-world dataset "OC3-HR" for multi-sourced Entity Linkage (Section 4).

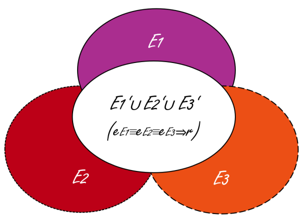

Figure 2: Visualisation of Scoping of multi-sourced Entity Collections based on Figure 1.

2 RELATED WORK

Traditional ER workflows consist of three sequential phases. In the **signature** phase, a numerical embedding strategy applies to all entity profiles. The vectorization of an entity profile, such as an attribute in a relational database named "customer address" can be based on tf-idf (Paulsen et al., 2023), the aggregation of pre-trained Word2Vec embeddings such as Glove or FastText (Cappuzzo et al., 2020). Further techniques combine words via sequence modeling, transformers, and self-supervised models (Brunner and Stockinger, 2020), (Thirumuruganathan et al., 2021), (Azzalini et al., 2021), (Zeakis et al., 2023).

The **blocking** phase generates a set of likely matching entity profiles into buckets. This phase rarely incorporates any further knowledge infusion except query tokens that vary with hyperparameter settings such as the size of the buckets and parallelization settings on hardware. Respective algorithms use dimensionality reduction techniques (PCA, t-SNE) to reduce the signature length. Then, nearest-neighbor algorithms (ENNs such as Hierarchical Clustering, K-Means, DBSCAN) (Azzalini et al., 2021) or approximate nearest-neighbor algorithms using hash indexing methods (ANNs such as LSH with FAISS or SCANN implementation) are applied to generate buckets of similar entities (Zeakis et al., 2023), (Johnson et al., 2019), (Guo et al., 2020).

Finally, the **filtering** phase constructs a set of verified linkages of matching entity profiles. This phase examines every inter-bucket pair and filters out unwanted linkages whose similarity is below a similarity threshold while keeping those that exceed the threshold. Filtering is optional and applies to the Cartesian set between entities within a blocking-generated

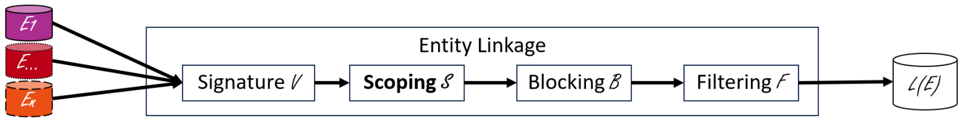

Figure 3: Entity Linkage Workflow with Scoping.

bucket (Koutras et al., 2021), (Traeger et al., 2022).

3 SCOPING

Scoping combines a ranking algorithm with a tunable threshold for generating streamlined entity collections. We assume a schema-aware, multi-sourced and unsupervised entity linkage environment. First, we provide an overview of the notations of this section in Table 1.

3.1 Scoping Definition

In entity linkage, we aim to find the set of linked entity profiles in which entities share the same real-world entity representation $\mathcal{L}(\mathcal{E}) = \{(e_a, e_b) : e_a \in \mathcal{E}_i$ and $e_b \in \mathcal{E}_j$ so that $e_a \equiv e_b \Rightarrow r\}$ where $\mathcal{E}_i, \mathcal{E}_j \in \mathcal{E}$ between entity collections.

Scoping utilizes the entity signatures v_i, the previously vectorized entity profiles processed by the Entity Signature phase $\mathcal{V}(\mathcal{E})$. Before applying the actual scoping method, an Entity Ranking algorithm $\mathcal{R}(\mathcal{V}(\mathcal{E}))$ computes an entity score for each entity signature, returning the tuple (e_i, s_i) where s_i is the score of entity profile e_i. The ranking algorithms presented in Section 3.2 categorize entity profiles with lower scores as linkable and higher scores as unlinkable in comparison to each other. The actual scoping algorithm $\mathcal{S}$, first, sorts the entity score tuples (e_i, s_i) in descending order so that $s_i < s_{i+1}$. Secondly, the algorithm filters the entity score tuples to identify and prioritize top-ranked entities with lower scores. We provide a single configurable threshold $p \in [0, 1]$ for the scoping algorithm that represents a radius (white overlapping space in Figure 2) for selecting linkable entities depending on the scores. The output of scoping generates a new subset $\mathcal{E}' \subseteq \mathcal{E}$ with fewer entity profiles selected from the original entity collection. Subsequently, blocking the entity profiles across $\mathcal{E}'$ instead of $\mathcal{E}$ results in higher quality entity pair candidates, resulting in less computational resources (space and time). Scoping generally differs in comparison to blocking in the sense that it aims to generate a subset entity collection $\mathcal{E}' \in \mathcal{E}$ without compromising the set of linkages $\mathcal{L}(\mathcal{E}') \equiv \mathcal{L}(\mathcal{E})$ regardless of the blocking sequence.

Figure 1 shows the effect of scoping on the original collection, in which a ranking algorithm computes a score for each entity and scopes 11 out of the 17 top-scored entities ($p = 0.65$). Unlinkable entities like "phone" and "country id" are omitted. With the scoped collections, we go down from 180 to 48 potential linkages without missing a single true linkage.

3.2 Ranking Methods in Scoping

In Section 3.1, we describe that the ranking methods $\mathcal{R}(\mathcal{V}(\mathcal{E}))$ process the entity signatures to compute an entity score tuple (e_i, s_i) used to scope a relative portion of $\mathcal{E}$ based on p in order to generate a subset of entity collections $\mathcal{E}'$. It is worth noting that the size of the data input in scoping is linear in the number of entity profiles $|\mathcal{E}_1| + |\mathcal{E}_{...}| + |\mathcal{E}_n|$ and not the Cartesian product (brute force) between all possible entity pairs between entity collections $|\mathcal{E}_1| \times |\mathcal{E}_{...}| \times |\mathcal{E}_n|$. Now, we present two modified outlier algorithms that compute anomaly scores v_i for each entity signature. Then, we introduce our novel encoder-decoder-based anomaly detection algorithms. We provide a computational complexity analysis for each of these algorithms that we evaluate in Section 4.

Z-Score: It is a statistical measure to quantify the entity's degree of dispersion. Its anomaly score implies how many standard deviations σ an entity signature differs from the mean μ of all entity signatures $(e_i, s_i) = ||\frac{(v_i - \mu)}{\sigma}||$. The Z-Score is computed per dimension of the entity signature and results in positive and negative floating values. Subsequently, the mean of the absolute (positive) dimension-based Z-scores along with the entity signature represents the entity score. The time complexity for computing the Z-scores is $O(|\mathcal{E}| \cdot \bar{v})$.

Local Outlier Factor (LOF): It is a density-based method that quantifies the local deviation of a data point from its neighborhood. The locality of the anomaly score depends on the degree of isolation (e.g., Euclidean or Cosine) between the entity signature and the surrounding entity signatures given by k-nearest neighbors. By comparing the local distances of one entity's signature to the local densities of its surroundings, those with substantially lower densities are considered to be outliers. We highlight that LOF requires k as a hyperparameter input representing the number of neighborhoods. The value of k directs the number of global linkages between entities and, therefore, highly influences the local density scores. The time complexity for computing the LOF is $O(|\mathcal{E}| \cdot \bar{v} \cdot k)$. We refer to the original paper (Breunig et al., 2000) for more details.

Encoder-Decoder: Next, we propose a novel method based on self-supervised encoder-decoder models to implement scoping. A main challenge in anomaly detection is the high dimensional space that occurs in the vectorized entities (signatures). In the past, anomaly detection with neural networks has received a lot of attention (Gong et al., 2019), (Bank et al., 2021), (Ruff et al., 2021), (Ilyas and Rekatsinas, 2022). The effect of learning a generative model and using it to detect anomalies has not yet been utilized within the Entity Linkage research space, and therefore, it motivates this work.

Once an encoder-decoder model is trained, frequent entities that are similar and appear to exhibit linkages pass the autoencoder with a lower reconstruction error. The reason for this is a bottleneck that appears in encoder-decoder algorithms, forcing the model to focus on recurring entities during the training rather than on rare anomalous ones. We adopt this criterion for identifying entities with a high reconstruction error as entities out of the linkage bound. To leverage an entity score of these methods for the scoping framework, a wrapped-up agent computes the mean squared error between the original and decoded entity signatures. We illustrate the functionality in Figure 4 and translate this model to an EL agent, as it provides feedback on the adaptability of entities for the linkage task based on a low or high reconstruction error. Additionally, the model of agents can be reused to rank entities of newly adapted collections. The benefit of using a trained model is that a holistic recomputation, such as needed for the Z-Score or LOF method, might not always be necessary. This approach can improve efficiency and save resources via

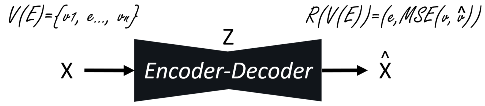

Figure 4: Agent for Entity Ranking in Scoping.

task transferability in EL pipelines.
Principal Component Analysis (PCA): Applying PCA onto the entity signatures transforms them into a lower-dimensional space. The reduced space should also capture relevant patterns. Each principal component in PCA quantifies the importance of the disparity of entities along its dimension based on linear hyperplanes. This is useful, as the signatures of unlinkable entities may exhibit unique patterns in high-dimensional space. PCA can be reused for scoping new entity collections by applying the model and comparing the mean squared error. We now present the PCA algorithm for scoping:

Data: $\mathcal{V}(\mathcal{E}) = (v_i)$ Entity Signatures
Result: $\mathcal{R}(\mathcal{V}(\mathcal{E})) = (e_i, s_i)$ Entity Scores
X = scaler(0,1).fit((v_i));
μ = mean(X);
//along dimensions;
pca = sklearn.PCA(nc).fit(X);
//singular value decomposition;
Z = pca.transform(X);
$\hat{X}$ = ($Z\cdot$ pca.components) + μ;
return $(s_i) \Leftarrow$ MSE(X,$\hat{X}$);

Algorithm 1: Entity Ranking $\mathcal{R}$ with **PCA**.

The first line casts an optional [0..1] normalization along the entity signature dimensions, transforming the set of entity signatures (v_i) to the input data set X. As entity signatures contain both negative and positive values along the dimensions, normalization simplifies the subsequent mean and similarity calculations. X is a $|E| \times \bar{v}$ matrix in which $|E|$ represents the number of entities of a collection and $\bar{v}$ represents the dimensional length of the entity

Table 1: Notation Table.

Notation	Meaning
e_i	Entity Profile
$\mathcal{E}_{source} = (e_1, e_{...}, e_n)$	Entity Collection
$(e_{\mathcal{E}_1} \equiv e_{\mathcal{E}_{...}} (\equiv e_{\mathcal{E}_n}) \Rightarrow r)$	Real-world Entity
$\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_{...} \cup \mathcal{E}_n$	Unified Entity Collection
$\mathcal{V}(\mathcal{E}) = (v_i)$	Entity Signature
$\mathcal{R}(\mathcal{V}(\mathcal{E})) = (e_i, s_i)$	Entity Ranking (score-tuple)
$\mathcal{S}((e_i, s_i), p) = \{(e_i, s_i) : \forall s_i < s_{i+1} \wedge \|(e_i, s_i)\| \times p\} = \mathcal{E}'$	Entity Scoping (threshold)
$\mathcal{B}(\mathcal{E}) = \{(e_a, e_b) : e_a \in \mathcal{E}_i \wedge e_b \in \mathcal{E}_j\}$ where $\mathcal{E}_i, \mathcal{E}_j \in \mathcal{E}$	Entity Blocking (brute-force)
$\mathcal{L}(\mathcal{E}) = \{(e_a, e_b) : e_a \in \mathcal{E}_i \wedge e_b \in \mathcal{E}_j, e_a \equiv e_b \Rightarrow r\}$ where $\mathcal{E}_i, \mathcal{E}_j \in \mathcal{E}$	Entity Linkage

signature. In the second step, we compute the μ vector of the mean along each dimension. Thirdly, we initialize PCA with the number of components given by the hyperparameter nc, subtract μ from each (normalized) vector signature X, and compute the singular value decomposition for each principal component with the same vector length as $\bar{v}$. Subsequently, we can now project the input data set X onto the lower-dimensional principal components, resulting in the encoded $|E| \times nc$ matrix we define as Z. Finally, we cast the decoder operation to generate $\hat{X}$. This involves reversing the dot-product between Z and principal components plus the entity-wise addition of the mean μ. Lastly, we compute the mean-squared error (MSE) between X and $\hat{X}$ and utilize these as the entity scores (s_i). The time complexity for PCA is $O(|\mathcal{E}| \cdot \bar{v}^2 + \bar{v}^3)$. We refer to the tutorial paper (Shlens, 2014) for more details on PCA.

Autoencoders: They are a special type of neural networks trained to encode data on a meaningful representation and reversely decode it back to its original state. These models are considered self-supervised as the data serves both as training input and output. Similar to PCA, we follow the assumption that a trained autoencoder learns relevant patterns more efficiently of normally distributed entity signatures but not for anomalous and unlinkable ones. Moreover, autoencoders with one latent layer and linear activation functions generalize PCA. In the following section, we provide a summary of autoencoders in the context of anomaly detection using the reconstruction error for scoping. More details on autoencoders can be found in the survey by (Bank et al., 2021).

We formally denote the encoder function as $A(\mathcal{V}(\mathcal{E}) = X) \Rightarrow Z$ mapping the set of the normalized entity signatures into a latent lower-dimensional representation. The decoder function $B(Z) \Rightarrow \hat{X}$ aims to transform the latent representation into the original input. Both the functions of A and B are trained over a number of epochs ep in order to minimize the mean reconstruction error converging to
$arg\ min_{A,B}[MSE(X, B(A(X)))]_{ep}$.

Contrary to PCA, normalization of the entity signatures not only simplifies the computation but allows the use of non-linear activation functions. Both the encoder and decoder functions can, therefore, construct more elegant and superior non-linear hyperplanes. At the same time, non-linear hyperplanes tend to overfit. For this reason, different types of regularization beyond the lower-dimensional bottleneck must be considered depending on the number of entities $|\mathcal{E}|$, signature length $\bar{v}$, and degree of deviations. Various possible configurations of the autoencoder may consider the network's depth or shallowness, the number of epochs, layers, neurons, activation functions, optimization algorithms, loss, and validation sampling configurations. The computational time complexity depends on those architectural choices; therefore, providing a $O()$ notation varies and is dependent on each different configuration. Due to the rising time complexity of backpropagating the weights of each neuron in each hidden layer over multiple epochs, we assume autoencoders to have a higher time complexity compared to the previously presented ranking methods. In the scoping context, we generally recommend preventing overfitting with regularization, as such a model would generate identical entity scores that are not useful for scoping.

4 EVALUATION

We evaluate the scoping approach on a real-world multi-sourced entity linkage dataset. We first present the performance metrics we use, then we describe the dataset, elaborate on chosen signature strategies, and the configuration of the ranking methods. All experiments were performed in Python Jupyter hosted by Google Collaboratory[1]. The dataset and code are available at `https://github.com/leotraeg/scoping`.

Performance Metrics: To measure the effectiveness of the algorithms for generating scoped entity collections $\mathcal{E}'$ from the original ones $\mathcal{E}$, we adopt typical metrics used in ER.

- *Reduction Ratio ($RR(\mathcal{E}', \mathcal{E})$)* reflects the time efficiency in scoping without relevance to the ground truth of linkages. It expresses the reduction in the number of entity comparisons between the scoped entity collections and the original ones:
$1 - ||\mathcal{B}(\mathcal{E}'_1, \mathcal{E}'_{...}, \mathcal{E}'_n)||/||\mathcal{B}(\mathcal{E}_1, \mathcal{E}_{...}, \mathcal{E}_n)||$.
- *Pair Completeness ($PC(\mathcal{E}', \mathcal{E})$)* estimates the number of potentially true entity linkages within the scoped entity collections with respect to the number of the ground truth entity linkages within the original entity collections:
$||\mathcal{L}(\mathcal{E}'_1, \mathcal{E}'_{...}, \mathcal{E}'_n)||/||\mathcal{L}(\mathcal{E}_1, \mathcal{E}_{...}, \mathcal{E}_n)||$.
- *Harmonic-Mean RR-PC ($HM(\mathcal{E}', \mathcal{E})$))* represents a combined metric between the two competing objectives of reduction ratio and pair completeness. $\frac{2 \cdot RR \cdot PC}{RR + PC}$.

The threshold p affects the collections of scoped entities $\mathcal{E}'$ in a major way. Knowing its value before-

[1]https://colab.research.google.com

Table 2: Description of OC3-HR: a multi-sourced entity linkage dataset with relational database schemas.

Entity Collections ($\mathcal{E}_A - \mathcal{E}_B$)	**#Tables**	**#Attributes**	$\lvert\mathcal{E}_A\rvert \times \lvert\mathcal{E}_B\rvert$	**#Attribute Linkages**
Domain-specific ∑OC3	**18**	**142**	**6617**	**47**
OC-Oracle – OC-MySQL	15(7+8)	102(43+59)	2537	19
OC-Oracle – OC-HANA	10(7+3)	83 (43+40)	1720	16
OC-MySQL – OC-HANA	11(8+3)	99 (59+40)	2360	12
Domain-agnostic ∑OC3-HR	**25**	**177**	**11587**	**62**
HR-Oracle – OC-Oracle	14 (7+7)	78 (35+43)	1505	0
HR-Oracle – OC-MySQL	15 (7+8)	94 (35+59)	2065	14
HR-Oracle – OC-HANA	10 (7+3)	75 (35+40)	1400	1

hand implies knowing the ground truth of entity linkages. As this is not true in reality, we propose to adjust p as an engineering task aiming to yield better performance by introducing two new metrics using the Area Under Curve and comparing them.

- *Area Under Curve PC APC*$(\mathcal{E}', \mathcal{E})$ evaluates the entity scoring utility in scoping. The more entity pairs that are correctly found with increasing p, the higher the single-valued APC metric. A higher APC for one ranking method allows more confidence in lowering the p threshold without time considerations.
- *Area Under Curve HM AHM*$(\mathcal{E}', \mathcal{E})$ helps to quantify the trade-off between the reduction ratio and pair completeness across all p thresholds. A higher AHM recommends a more robust scoping approach considering both the pair completeness and time efficiency.

Schemas and Experiments: The datasets that we use contain only schema information from Oracle, MySQL, and SAP HANA, without instance data. First, we perform a set of experiments on a domain-specific set of order-customer (OC) schemas with 47 true inter-schema attribute linkages out of 6617 attribute-pair candidates. Then, we conduct the same experiments on a domain-agnostic set of schemas by extending the domain-specific schemas with a human resources (HR) schema. We annotate 15 additional inter-schema attribute pairs since the OC-MySQL schema contains attribute linkages between employees and offices. Consequently, the domain-agnostic setting contains 11587 attribute pair candidates, of which 62 are considered true. We provide a detailed summary of the dataset in Table 2.

Signature: We uniformly preprocess the textual descriptions of each entity across all collections by concatenating the table and attribute names, splitting concatenated words, and removing repetitive words. Based on the comparative analysis by (Zeakis et al., 2023), we aggregate static Glove embeddings trained on Common Crawl without out-of-vocabulary vector retrievals (Pennington et al., 2014) and use Sentence Transformer Bert (gtrt5-base) (Reimers and Gurevych, 2019) as the best dynamic strategy.

Ranking: We use the following configurations for the ranking methods:

- Z-Score: We use the default implementation of the SciPy[2] library.
- LOF: We import the sklearn neighbors library[3] and specify the number of neighbors $k = 15$ as these are the average number of linkages between the entity collections.
- PCA: We use the Lapack SVD implementation of the sklearn decomposition library[3] and use $nc = 2$ (number of principal components) as we deal with a small size of $|E|$.
- AE: We use the Keras library and configure an autoencoder with three intermediate layers to extend the network complexity to PCA. We use rectified linear units (ReLUs), Adam as the optimizer, the mean squared error (MSE) as the loss function, and a shuffled test-train split of 20%. To prevent identity functions, we define the small size of ten epochs (fixed early stopping) but initialize the model ten times and sum up each entity's MSE to stabilize the final entity score.
- Ensemble: We take the mean score across the normalized entity score tuples (e_i, s_i) of the best-performing configurations of Z-Score, LOF, PCA, and AE based on APC, AHM$_{\mathcal{B}}$, and AHM$_F$. In this regard, an equally weighted ensemble of entity scores works similarly to a random forest (supervised) or consensus clustering (unsupervised).

Blocking: Blocking primarily affects the time reduction ratio, while scoping affects the completeness of entity linkage pairs. To show this distinction, we employ two blocking modules: the first is a simplistic entity blocking module $\mathcal{B}$ that schedules all potential inter-source-linkages (Cartesian product) between the (scoped) entity collections at point p. The second one

[2]https://docs.scipy.org/doc/scipy

[3]https://scikit-learn.org

Table 3: Scoping configurations and Area Under Curve PC and HM performances on OC3-HR dataset.

Scoping			Domain-specific			Domain-agnostic		
$\mathcal{V}$ Signature	$\mathcal{R}$ Ranking	$\mathcal{R}$ Parameter	APC	$AHM_{\mathcal{B}}$	AHM_F	APC	$AHM_{\mathcal{B}}$	AHM_F
Glove	Z-score	*none*	52.36	**47.30**	53.69	**54.21**	44.81	53.46
gtrt5	Z-score	*none*	**60.52**	46.82	**54.11**	50.57	**45.06**	**53.61**
Glove	LOF	$k = 15$	54.37	45.69	52.53	52.62	44.60	53.82
gtrt5	LOF	$k = 15$	**61.59**	**48.19**	**55.40**	**57.69**	**46.91**	**56.20**
Glove	PCA	$n = 2$	53.03	45.38	52.13	53.61	45.33	54.32
gtrt5	PCA	$n = 2$	**61.80**	**49.06**	**56.17**	**56.81**	**46.94**	**55.79**
Glove	AE	610,300,10,300,610	52.69	47.33	53.83	49.97	44.95	53.73
gtrt5	AE	778,300,10,300,778	**64.05**	**49.45**	**56.73**	**57.58**	**47.11**	**56.02**
	Ensemble	$\max_{APC,AHM_{\mathcal{B}},AHM_F}$	63.86	**50.92**	**58.10**	**58.81**	**47.45**	**56.38**

is the efficient locality-sensitive hashing-based similarity search blocking module F implemented with the Python package FAISS[4] (Papadakis et al., 2022) (Paulsen et al., 2023), (Zeakis et al., 2023). This blocking scheme queries each entity signature and outputs a maximum of $k = 50$ linkage candidates based on the L2-distanced nearest neighbors.

Scoping Results and Discussion: We evaluate the scoping approach on the OC3-HR dataset. We report the AUC metrics for the pair completeness (APC) and the harmonic mean for the brute force ($AHM_{\mathcal{B}}$) and FAISS-based ($AHM_{\mathcal{F}}$) blocking modules. The results are based on the entity signature $\mathcal{V}$, ranking method $\mathcal{R}$, and parameter configurations summarized in Table 3. The best result per ranking method is highlighted in bold. Figure 5 and 6 plot the best-performing stand-alone and ensemble configurations for APC and AHM with the performance (y-axis) of the time reduction ratio (yellow), pair completeness (blue), and the harmonic mean (green) on the increasing relative threshold parameter p (x-axis). Generally, Sentence Transformer Bert (gtrt5) signatures outperform word2vec (Glove) with minor exceptions for the Z-Score method. It is worth noting that we also compared two blocking methods, showing that the FAISS-based one improved in computational time reflected in AHM_F. However, none of them had any effect on the number of true linkages as measured in APC.

We first discuss the results obtained for the domain-specific dataset. The best-performing stand-alone model for APC and AHM is the autoencoder with gtrt5 signatures. We explain the 2.25% domain-specific APC improvement for autoencoders with the ensembling training nature: autoencoders' compression and decompression functions are trained over multiple epochs with a shuffled train-test split of entity signatures. On the contrary, the ensemble method yields the best performance for AHM.

We now discuss the results obtained for the domain-agnostic dataset. Among the signature strategies, all ranking methods and configurations perform the best with gtrt5 signatures except for Z-Score in APC. The best stand-alone APC performance is achieved by LOF, falling 1% below the ensemble. For the domain-agnostic dataset, the ensemble method performs best in both AHM and APC. In general, the domain-agnostic dataset contains several pairs that are not linkable due to the fact that the schemas do not reflect similar domains. Although the true linkages are still captured in the domain-agnostic schemas, it is expected to observe a decrease in the performance metrics due to the dissimilarity of the schemas.

Summary: Evaluations on multi-sourced schemas show that autoencoders with gtrt5 signatures perform best in the domain-specific entity linkage task. We highlight that this scoping configuration reduces the search space to 77% of entities and still contains all linkages. The impact of the ranking method is more relevant for the domain-specific setting rather than the domain-agnostic setting. Generally, ensembling the entity scores of different ranking methods can yield more robust results for both APC and AHM. Finally, all scoping configurations for both the domain-specific and agnostic settings intersect with the graph of the time reduction ratio at 50% (x-axis) and the pair completeness at around 75% (y-axis). This means that scoping can reduce the original set of entity collections when the time for comparisons is limited and still retain a high portion of true linkages.

5 CONCLUSION

This paper introduced scoping, a new phase in the EL pipeline that ranks entities and generates a subset $\mathcal{E}'$ of the raw entity collections $\mathcal{E}$ while leaving intact the major set of true linkages. We have shown that

[4] https://github.com/facebookresearch/faiss

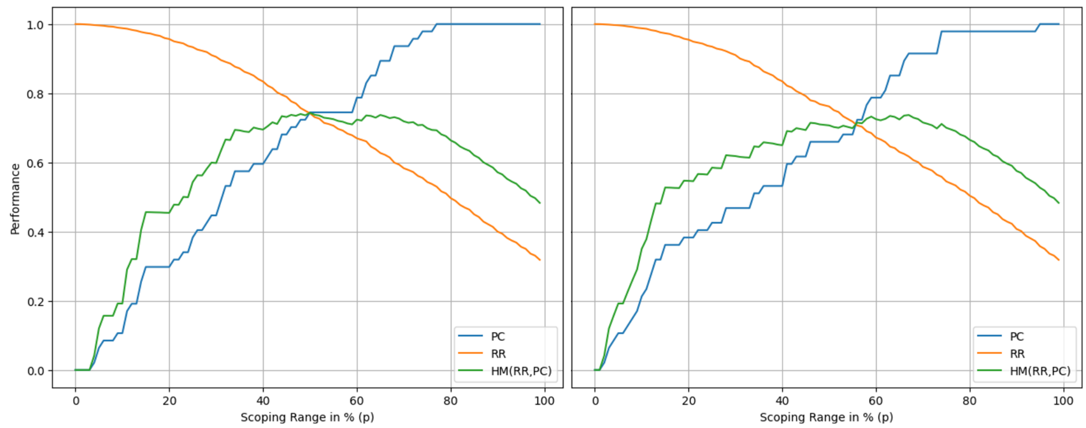

Figure 5: RR, PC, and HM performance on domain-specific OC3 dataset (left: AE with gtrt5 and right: ensemble).

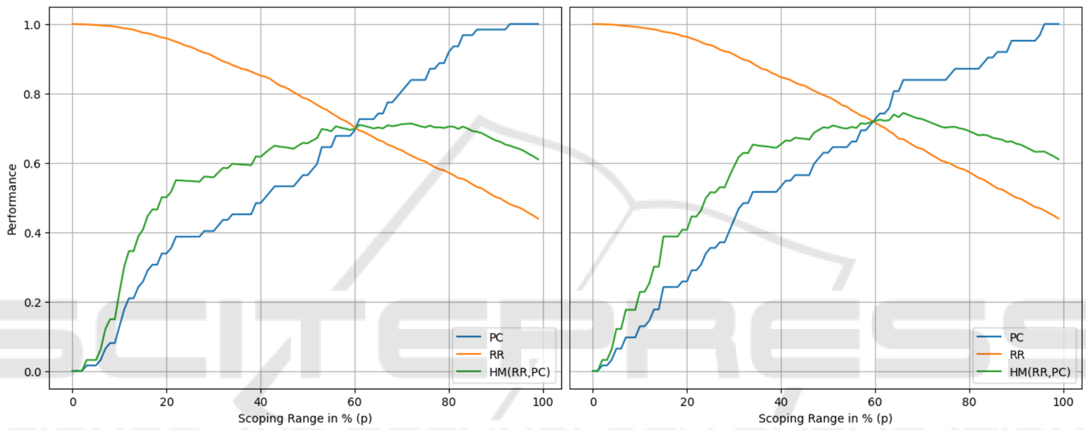

Figure 6: RR, PC, and HM performance on domain-agnostic OC3-HR dataset (left: LOF with gtrt5 and right: ensemble).

models learning to compress and decompress entities from multiple data sources can be used to scope out linkable entities with better or almost equal performance compared to existing ranking methods. We see the various autoencoder network configurations as a strength to better adapt to different numbers of entities $|\mathcal{E}|$, the dimensionality of entity signatures $\bar{v}$, and the degree of domain specificity. Moving on to the potential advantages of self-supervised models, we highlight that PCA and the autoencoder model can be reused to scope new incoming entity collections. In future work, we plan to investigate autoencoders enriched with a multi-modal network to incorporate textual descriptions and instances from entity profiles.

REFERENCES

Azzalini, F., Jin, S., Renzi, M., and Tanca, L. (2021). Blocking Techniques for Entity Linkage: A Semantics-Based Approach. *Data Science and Engineering*, 6(1):20–38.

Bank, D., Koenigstein, N., and Giryes, R. (2021). Autoencoders. arXiv:2003.05991 [cs, stat].

Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. (2000). LOF: identifying density-based local outliers. *ACM SIGMOD Record*, 29(2):93–104.

Brunner, U. and Stockinger, K. (2020). Entity matching with transformer architectures - a step forward in data integration. Medium: application/pdf Publisher: OpenProceedings.

Cappuzzo, R., Papotti, P., and Thirumuruganathan, S. (2020). Creating Embeddings of Heterogeneous Relational Datasets for Data Integration Tasks. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, pages 1335–1349, New York, NY, USA. Association for Computing Machinery.

Gong, D., Liu, L., Le, V., Saha, B., Mansour, M. R., Venkatesh, S., and Hengel, A. v. d. (2019). Memorizing Normality to Detect Anomaly: Memory-augmented Deep Autoencoder for Unsupervised Anomaly Detection. arXiv:1904.02639 [cs].

Guo, R., Sun, P., Lindgren, E., Geng, Q., Simcha, D., Chern, F., and Kumar, S. (2020). Accelerating large-scale inference with anisotropic vector quantization.

Ilyas, I. F. and Rekatsinas, T. (2022). Machine Learning and Data Cleaning: Which Serves the Other? *Journal of Data and Information Quality*, 14(3):13:1–13:11.

Johnson, J., Douze, M., and Jégou, H. (2019). Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Koutras, C., Siachamis, G., Ionescu, A., Psarakis, K., Brons, J., Fragkoulis, M., Lofi, C., Bonifati, A., and Katsifodimos, A. (2021). Valentine: Evaluating Matching Techniques for Dataset Discovery. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 468–479. ISSN: 2375-026X.

Lerm, S., Saeedi, A., and Rahm, E. (2021). Extended Affinity Propagation Clustering for Multi-source Entity Resolution. *BTW 2021*.

Papadakis, G., Fisichella, M., Schoger, F., Mandilaras, G., Augsten, N., and Nejdl, W. (2022). How to reduce the search space of Entity Resolution: with Blocking or Nearest Neighbor search? arXiv:2202.12521 [cs].

Paulsen, D., Govind, Y., and Doan, A. (2023). Sparkly: A Simple yet Surprisingly Strong TF/IDF Blocker for Entity Matching. *Proceedings of the VLDB Endowment*, 16(6):1507–1519.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Ruff, L., Kauffmann, J. R., Vandermeulen, R. A., Montavon, G., Samek, W., Kloft, M., Dietterich, T. G., and Müller, K.-R. (2021). A Unifying Review of Deep and Shallow Anomaly Detection. *Proceedings of the IEEE*, 109(5):756–795. arXiv:2009.11732 [cs, stat].

Shlens, J. (2014). A Tutorial on Principal Component Analysis. arXiv:1404.1100 [cs, stat].

Thirumuruganathan, S., Li, H., Tang, N., Ouzzani, M., Govind, Y., Paulsen, D., Fung, G., and Doan, A. (2021). Deep learning for blocking in entity matching: a design space exploration. *Proceedings of the VLDB Endowment*, 14(11):2459–2472.

Traeger, L., Behrend, A., and Karabatis, G. (2022). Inteplato: Generating mappings of heterogeneous relational schemas using unsupervised learning. In *2022 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 426–431.

Zeakis, A., Papadakis, G., Skoutas, D., and Koubarakis, M. (2023). Pre-Trained Embeddings for Entity Resolution: An Experimental Analysis. *Proceedings of the VLDB Endowment*, 16(9):2225–2238.