TableDC: Deep Clustering for Tabular Data

HAFIZ TAYYAB RAUF, Department of Computer Science, University of Manchester, UK
ANDRÉ FREITAS, Department of Computer Science, University of Manchester, UK and IDIAP Research Institute, Switzerland
NORMAN W. PATON, Department of Computer Science, University of Manchester, UK

Deep clustering (DC), a fusion of deep representation learning and clustering, has recently demonstrated positive results in data science, particularly text processing and computer vision. However, joint optimization of feature learning and data distribution in the multi-dimensional space is domain-specific, so existing DC methods struggle to generalize to other application domains (such as data integration). In data management tasks, where high-density embeddings and overlapping clusters dominate, a data management-specific DC algorithm should be able to interact better with the data properties to support data integration tasks. This paper presents a deep clustering algorithm for tabular data (TableDC) that reflects the properties of data management applications that cluster tables (schema inference), rows (entity resolution) and columns (domain discovery). To address overlapping clusters, TableDC integrates Mahalanobis distance, which considers variance and correlation within the data, offering a similarity method suitable for tabular data in high-dimensional latent spaces. TableDC also shows higher tolerance to outliers through its heavy-tailed Cauchy distribution as the similarity kernel. The proposed similarity measure is particularly beneficial where the embeddings of raw data are densely packed and exhibit high degrees of overlap. Data integration tasks may also involve large numbers of clusters, which challenges the scalability of existing DC methods. TableDC learns data embeddings with a large number of clusters more efficiently than baseline DC methods, which scale in quadratic time. We evaluated TableDC with several existing DC, Standard Clustering (SC), and state-of-the-art bespoke methods over benchmark datasets. TableDC consistently outperforms existing DC, SC and bespoke methods.

CCS Concepts: • **Information systems → Data cleaning**; **Entity resolution**.

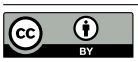
Additional Key Words and Phrases: Data integration, deep clustering, data cleaning, Mahalanobis distance

ACM Reference Format:
Hafiz Tayyab Rauf, André Freitas, and Norman W. Paton. 2025. TableDC: Deep Clustering for Tabular Data. *Proc. ACM Manag. Data* 3, 3 (SIGMOD), Article 229 (June 2025), 28 pages. https://doi.org/10.1145/3725366

1 Introduction

Deep Clustering (DC) combines the unsupervised grouping of related data items with the learning of clustering-friendly representations for the data to be clustered. DC uses deep neural network architectures with unsupervised clustering mechanisms to effectively classify multi-dimensional, unlabeled data. Early implementations primarily used autoencoders for dimensionality reduction, where the embeddings in the latent space were clustered using standard clustering (SC) algorithms such as K-means [87]. Advances in DC have led to more sophisticated architectures, based on variational autoencoders [20], Generative Adversarial Networks [58], and self-supervised learning

Authors' Contact Information: Hafiz Tayyab Rauf, Department of Computer Science, University of Manchester, Manchester, UK, hafiztayyab.rauf@manchester.ac.uk; André Freitas, Department of Computer Science, University of Manchester, Manchester, UK and IDIAP Research Institute, Martigny, Switzerland, andre.freitas@manchester.ac.uk; Norman W. Paton, Department of Computer Science, University of Manchester, Manchester, UK, norman.paton@manchester.ac.uk.



ACM 2836-6573/2025/6-ART229
https://doi.org/10.1145/3725366

paradigms [2]. Recent DC methods focus on integrating representation learning and cluster assignment within end-to-end trainable models using graph-based neural networks [6, 53, 80] to capture relational structures within data.

DC has been employed successfully in several domains, particularly image processing [67, 97, 97], which includes downstream tasks such as segmentation and object detection [97], clustering of in and out-of-distribution noise in corrupted images [1], and unsupervised image alignment and clustering [39, 67]. Further typical applications of DC include community and anomaly detection [25, 51] and medical applications [40]. These DC proposals excel in determining complex patterns within images and graphs, extracting features, and subsequently clustering similar images and communities based on those patterns. The successful application of DC in specific domains has led to the development of more specialized algorithms that reflect specific properties of the domain features, even if the algorithms are not explicitly domain-specific. For example, in [15], the authors use domain-specific 3D geometric consistency to guide the learning of 2D image representation and solve several 2D-image-based downstream tasks.

In data management, and in particular data integration and cleaning, a variety of important problems make use of clustering, including table clustering for *schema inference* [8, 43], row clustering for *entity resolution* [18, 35, 63] and column clustering for *domain discovery* [62, 66]. A previous study investigated the application of existing DC algorithms to data integration problems, with positive results, in that DC algorithms consistently outperform established SC algorithms for a variety problems [70]. However, we observe that existing DC algorithms often fail to take into account the fact that tabular data is fundamentally different from that in image processing applications, for which existing algorithms have often been developed. In this paper, we propose a new DC algorithm (Table DC) for tabular data and show that it consistently outperforms existing SC and DC algorithms.

In particular, TableDC reflects the properties of table embeddings and data management applications, specifically: (i) Data representations are typically dense, where features are closely packed together. As the dimensionality increases, it becomes more challenging for the models to interpret the relationships between features. For example, clustering the records of two persons with the same *name* and *date of birth* but different *addresses* and *emails* becomes more complex as the clustering algorithm struggles to decide which features to prioritize. High density in data reduces the differences in features and makes it difficult to differentiate between clusters effectively. The closeness of data points in the latent space leads to ambiguity, where a data point could reasonably belong to multiple clusters. (ii) Tabular data contains higher semantic and syntactic variability levels than image data, leading to increased ambiguity and cluster overlap. For example, tabular data can have varied formats for dates *("05/05/2021" vs. "May 5, 2021")* and differing labels for the same concept *("New York" vs. NY)*, while image data typically has more consistent pixel values across similar images. Tables, columns and rows commonly involve the composition of different types of semantic objects (e.g., named entities, categories, numbers), each in their own way subject to ambiguity, leading to potentially overlapping embeddings when projected to high-dimensional latent spaces. In contrast, while complex, image data is based on a single data modality, with the expectation of smoother transfer of visual features across domains, naturally conforming to a better cluster separation. (iii) In a data management setting, data can have a large number of clusters compared to existing DC applications. In previous work [70], it was observed that existing DC algorithms struggle with scalability when the data has a large number of clusters. This affects the DC method's performance in terms of the high probability of misclassifications, a higher computational burden, and cluster instability, where small changes in data and model parameters lead to significant changes in cluster assignments.

The contributions of this paper are:

(1) The identification of features of data management applications that challenge the assumptions that underpin existing DC algorithms.
(2) The proposal of a new DC algorithm, TableDC, that builds on the analysis at (1) to provide an algorithm that: (i) Maintains a balance between cluster compactness and separation during cluster center initialization to better assist TableDC training with robust cluster shapes. (ii) Integrates Mahalanobis distance, a covariance-aware distance that considers the data's variance and covariance to handle dense features, unlike Euclidean distance, which treats all dimensions equally. (iii) Offers flexibility in overlapping clusters through its heavy-tailed Cauchy distribution as a similarity kernel, especially when the cluster boundaries are ambiguous. (iv) Scales in quasi-linear time under significant increases in the number of clusters.
(3) The experimental evaluation of TableDC from (2) with both state-of-the-art SC and DC algorithms and recent bespoke algorithms on real-world datasets for table clustering (schema inference), row clustering (entity resolution) and column clustering (domain discovery), which shows that TableDC significantly outperforms existing techniques.

2 Related Work

This section reviews related work in areas of relevance to the development of a deep clustering algorithm for data management applications, specifically: (i) *existing DC algorithms, their features and applications;* and (ii) *clustering in data management*, focusing on the three applications on which we evaluate techniques and representations for tables, columns and rows.

2.1 Existing DC algorithms

The basic DC architecture consists of two phases: (i) representation learning and (ii) clustering. The joint optimization of both phases helps each to improve the algorithm's performance iteratively. A precursor to (i) and (ii) is (iii) cluster initialization, which contributes to the algorithm's performance by providing the initial data shape. With better cluster initialization, a DC algorithm is less likely to get stuck in local minima, and requires fewer converging iterations.

Regarding (i), the most widely used architecture is the basic Auto-encoder (AE) [87], where the encoder function f_e encodes an input representation x_i into a latent representation given by $h_i = f_e(x_i) = \frac{1}{1+e^{-(Wx_i+b_i)}}$. Similarly, the decoder function f_d decodes h_i back into the reconstructed input $x'_i = f_d(h_i)$. Here, W and b_i represent the weights and biases of the neural network, respectively. The optimization function of AE with N samples is defined as $f_{min} = \min \frac{1}{N} \sum_{i=1}^{N} \|x_i - x'_i\|^2$. Many recent DC proposals use generative models in representation learning, especially Variational Auto Encoders (VAEs), which combine an AE architecture with probabilistic graphical models. In VAEs, the encoder maps the input data into a probabilistic distribution over the latent space while the decoder reconstructs the data from the latent probabilistic representation. Some recent proposals with VAEs in representation learning are DGG [88], LTVAE [49], VaDE [38] and MFCVAE [27].

To benefit from the local and global structure of data in representation learning, several recent DC proposals combine basic AE learning with a Graph Convolutional Network (GCN) to assist the model in learning the structural information. Some recent examples are DCRN [53], DFCN [80] and SDCN [6]. Subspace representation learning is another recent technique that identifies subspaces that optimize cluster separation, unlike AEs and VAEs, which require additional clustering mechanisms. Subspace learning seeks to identify and represent underlying subspaces within high-dimensional data, while AEs and VAEs focus on reconstructing the input data as accurately as possible. The latest DC methods based on Subspace representation learning are EDESC [9], DMCAG [19], SAGSC [29] and OMSC [14].

DC is inherently more complex for the tabular data embeddings obtained by Language Models than widely used sparse image or textual datasets like CIFAR-10 [44], CIFAR-100 [44], STL-10 [17], ImageNet [74], USPS [45] and Reuters [47], primarily due to data structure and feature representation differences. Image datasets include spatial and visual coherence that DC methods can better differentiate, even when the resolution is low or the number of classes is high. DC has been used widely on complex and varied datasets unlike ImageNet, where, despite the vast diversity, the visual patterns (such as extracted from different textures, shapes, objects and backgrounds) retain a level of consistency

On the other hand, the primary issue with data management datasets is the high density and significant overlap in the feature space of their embeddings; specifically, tabular data embeddings represent a complex relationship among different features, including categorical and numerical data.

In relation to (ii), clustering, the aim of existing work on DC is to use the low dimensional representation from the representation learning module and obtain clustering assignment probabilities for soft clustering. In the related work, several clustering mechanisms are employed to jointly work with representation learning for optimal clustering. The most widely used one is self-supervised clustering [6, 21, 33, 65, 80, 86], which aims to group data points based on learned representations without explicit labels; the approach uses two distributions: the cluster assignment distribution Q representing the probability that a data instance belongs to a particular cluster, and an auxiliary distribution P which is a modified version of Q that emphasizes more confident cluster assignments and minimizing the Kullback-Leibler (KL) divergence between Q and P. Q is calculated using the distance between the data point x_i and each cluster centroid μ_k. The closer x_i is to μ_k, the higher the probability Q. Self-supervised clustering focuses on high-confidence instances due to the squaring of probabilities in P and preventing cluster dominance by normalizing the probabilities, ensuring balanced clustering assignments.

Pseudo-labeling is another clustering technique recently utilized in DC [61, 97], where the model generates labels that are then used to fine-tune the model, creating a feedback loop where the model iteratively refines its clustering assignment. The success of contrastive learning has led to its use in DC [54, 55, 78, 96], where data points from low dimensional representations are paired such that the same cluster (positive pairs) is contrasted with the different clusters (negative pairs). Further improvements in contrastive deep clustering include the contrast comparison between instance-to-instance, instance-to-cluster, and cluster-to-cluster [97].

TableDC employs a self-supervised clustering approach by comparing the encodings of tables, rows or columns within a latent space. The selection of an appropriate distance function is critical in this context, as it determines how the distances between each table, row or column and the cluster centroids in the latent space are measured. A simple distance metric like Euclidean distance is beneficial when dealing with data where features exhibit limited variance and correlation, as in image vectors or sparse data matrices [6, 80]. However, a more sophisticated approach is required in dense embeddings, where the data points show significant feature correlation. Here, the Mahalanobis distance is particularly advantageous [56]. Its ability to account for feature correlation enables the model to discern even subtle differences among clusters, mainly when a cluster shows high similarity across multiple features. Mahalanobis distance is particularly relevant when comparing tables, columns or rows, as it allows for a more nuanced interpretation and grouping of data points based on their inherent relationships and similarities.

In relation to (iii), most existing works use K-means to initialize clusters, whether self-supervised or subspace clustering such as EDESC [9], DCRN [53], DFCN [80] and SDCN [6]. The quality of the initial clusters significantly affects the final clusters. To our knowledge, there is no related work

on different cluster initialization approaches in the DC environment. To address this, we compare different cluster initialization approaches in an ablation study in Section 5.1.

2.2 Clustering in data management

Clustering is important in data management for several data integration and cleaning tasks, including those that apply on tables, rows and columns, as discussed here.

Schema inference is the process of identifying datasets that share structural features and, based on these recurring features, of generating a schema that can represent the data in the underlying datasets [41, 64]. Schema inference can generate a schema that fully describes some closely related datasets [3, 4] or summarize structures that appear in more diverse datasets [85, 89, 92]. A common first step in the latter case is the identification of recurring structural patterns, that can be used as types in an inferred schema. For example, existing tables *Employer* and *Company* may both have overlapping attributes that describe features like the *name*, *type*, *location*, *number of employees* and *turnover* of an organization, and could be candidates to be associated with a single entity type in an inferred schema. However, the existing *Employer* and *Company* tables are unlikely to be identical, and thus some notion of similarity needs to be used to identify which tables can be represented by a single entity type in an inferred schema. In this setting, a variety of SC algorithms have been used to group datasets with similar structures, for example looking for similar attribute names or values in underlying datasets (e.g., [8, 43, 71, 79, 82]). In this paper, we cluster tables based on their similarity, where each cluster of tables is a candidate entity type for the inferred schema.

Entity Resolution is the well-explored problem of identifying two or more records that represent the same real-world object [16]. Although many entity resolution proposals focus on pairwise comparisons, some proposals incorporate clustering because the transitive closure of pairwise similarity may not always lead to robust clusters [18, 22, 30, 35, 57, 75]. There has also been widespread use of deep representation learning for entity resolution (e.g. [23]), including the use of pre-trained language models (e.g. [50, 94]); in this paper, we show how such pre-trained representations can be fine-tuned for clustering rows without recourse to labelled data.

Domain Discovery involves identifying columns that draw values from the same collection of values. We cast domain discovery as a clustering problem, where the goal is to cluster a set of columns that share semantic types/domains even if they use different terminologies, structures and meta-data. For example, a product with the property *"aspect ratio of an image"* in different *eCommerce* sources may be described as a column with the header*"Aspect Ratio"*, *"Image Ratio"*, *"Image Ratio WH"*, *"Still Picture Aspect Ratios"* or *"Supported Aspect Ratio"* in different datasets, and the columns may contain a variety of values. Although these representations differ, they all describe the same domain. Most existing works have used bespoke algorithms [48, 62, 66] for domain discovery. RaF-STD [66] uses clustering to discover semantic types for heterogeneous sources. Similarly, D4 [62] provides a column-based clustering approach to discovering local and strong domains from a set of columns. Building on language models, Starmie [28] provides column clustering based on column representations obtained through contrastive learning. In this paper, like Starmie, we use self-supervised techniques to fine-tune column representations, though unlike in Starmie those representations are specifically targeted to clustering.

3 Proposed Model

3.1 Architecture

We propose a clustering algorithm, TableDC, that benefits from representation learning in an end-to-end framework for clustering all of tables, rows and columns; the primary components of TableDC

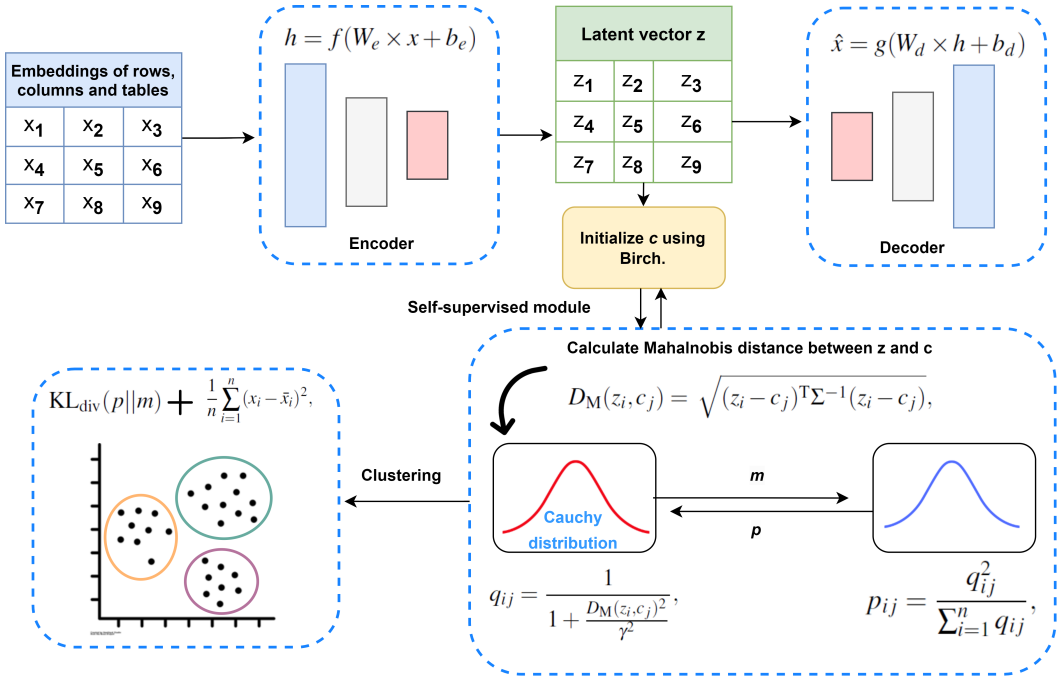


Fig. 1. Overview of TableDC framework. An autoencoder takes an embedding matrix as input, representing a set of tables, rows or columns, as required by the application. z is the latent representation obtained from the autoencoder, and c is initialized using Birch. In the self-supervised module, the distribution m is calculated from q by taking the Mahalanobis distance between z and c and setting the Cauchy distribution as a kernel to measure the similarity. p is the target distribution, which is also calculated from q.

are illustrated in Figure 1. The objective in deep clustering is to identify a latent representation (z in Figure 1) that both retains key features of the input embeddings and is suitable for clustering.

3.1.1 Autoencoder. An autoencoder consists of two main parts: the encoder and the decoder. The encoder function compresses the input x (an embedding matrix generated through embedding models that takes as input tables, rows or columns) into a latent-space representation. Mathematically, it can be represented as:

$$h = f(W_e \times x + b_e) \tag{1}$$

where W_e is the weight matrix, b_e is the bias term, and f is an activation function for the encoder, such as sigmoid or ReLU [59]. The result, h, is the encoded representation of the input x. The decoder function works to reconstruct x from the internal representation. It aims to map the encoded data back from the latent space to the original space. The decoder can be represented as:

$$\hat{x} = g(W_d \times h + b_d) \tag{2}$$

where h is the encoded representation from the encoder, W_d is the weight matrix, b_d is the bias term, and g is an activation function for the decoder. The result, $\hat{x}$, is the reconstructed representation of the original input x.

3.1.2 Mahalanobis Distance in self-supervised module. The self-supervised module aims to learn embeddings for clustering dense and overlapping data. It uses the Mahalanobis distance [56] to capture variance and correlations in the tables, rows or columns. Mahalanobis distance measures the

distance between data points and their centroids as it models the variance of different dimensions (in the context of high-dimensional tabular data) and is less sensitive to noise (outliers in tabular data that can distort clustering). Mahalanobis distance prevents features from disproportionately impacting the distance measure due to their scale. For example, in entity resolution tasks, embeddings of attributes like *Song Length* represented as *03:39* (minutes: seconds) in one source and *219* (seconds) in another. The vector representations for the two instances will be different. Without considering the scale for both attributes, the representation in seconds could disproportionately influence the distance calculation over other attributes. Mahalanobis distance normalizes these variations in scale so that all features contribute equally.

Unlike learning a Euclidean distance in a self-supervised module [6], which assumes that all dimensions of the data are orthogonal for the clustering, the Mahalanobis distance shows some degree of correlation between different dimensions by weighting the importance of all dimensions, which could be different based on their semantics. The importance of dimensions is weighted by scaling distances using the covariance matrix, which captures variance and relationships between dimensions.

Assume we have a latent vector with n data points $z = z_1, z_2, \ldots, z_n$, where each z_i represents a row, column or table. The goal is to cluster the data points into $\mathbb{K}$ clusters with centers $c = c_1, c_2, \ldots, c_k$. To compute the Mahalanobis distance between z and c, a covariance matrix Σ is required. Given the noisy nature of the data, such as columns with missing values, tables with different unit measurements, and duplicate rows, the covariance matrix must mitigate the influence of noise. The Mahalanobis distance is computed as:

$$D_{\mathrm{M}}(z_i, c_j) = \sqrt{(z_i - c_j)^{\mathrm{T}} \Sigma^{-1} (z_i - c_j)}, \tag{3}$$

where Σ is the covariance matrix, z_i represents a latent data point, and c_j represents a cluster center. We used Cholesky decomposition to simplify the inversion of the covariance matrix Σ^{-1}. It factorizes Σ into the product of a lower triangular matrix L and its transpose L^T, as expressed below:

$$C = LL^T, \tag{4}$$

where L is the lower triangular matrix with real and positive diagonal entries, and L^T is its transpose. The Cholesky decomposition efficiently computes Σ^{-1} by solving a linear system:

$$\Sigma^{-1} = (C)^{-1} = L^{-T} L^{-1}. \tag{5}$$

To address the noisy nature of the data, such as missing values, varying units, and duplicate rows, we scale the identity matrix with a factor $\delta = 0.01$ to ensure that the covariance matrix $\Sigma = \delta \cdot I$ remains non-singular while reducing the noise effect. This scaling manages feature variances, prevents multicollinearity, and stabilizes clustering when clusters overlap or when noisy correlations exist in the data.

3.1.3 Cauchy Distribution as similarity kernel. In clustering applications, the similarity kernel measures the similarity between z_i and c_j. The Student's t-distribution [81] is a common choice for existing DC applications as a kernel to measure the similarity between z_i and c_j. However, due to its large degree of freedom parameter, the Student's t-distribution has less tolerance for outliers and overlapping clusters, as it closely approximates a normal distribution. In a data integration setting, the data is more prone to overlaps, i.e., the same magnitude with different semantics. To address this, we integrate the Cauchy distribution [11], which has an undefined mean and variance.

The Cauchy distribution is resilient to outliers as its shape remains unaffected by extreme data points. For example, in clustering item prices such as £2, £4, and an outlier of £500, the Cauchy distribution reduces the outlier's influence by assigning it a low similarity score and preserves the primary cluster boundaries. Similarly, in clustering semantically overlapping features such as *Medical Center*, *Hospital Name* and *Surgery*, the Cauchy distribution checks overlapping vectors through its heavy-tailed properties without applying overly strict boundaries. In TableDC, the soft assignments q of data points to clusters are computed using the *Cauchy distribution*, defined in terms of the Mahalanobis distance $D_{\mathrm{M}}(z_i, c_j)$ between a data point z_i and a cluster center c_j. The soft assignment q_{ij} for a data point z_i to a cluster j is calculated as:

$$q_{ij} = \frac{1}{1 + \frac{D_{\mathrm{M}}(z_i, c_j)^2}{\gamma^2}}, \tag{6}$$

where γ is a hyperparameter for the Cauchy distribution. The optimal value of γ is 0.5, which was determined through grid search. We then normalize q to ensure that the sum of the soft assignments for each data point is 1:

$$q = \frac{q}{\sum_{j=1}^{k} q_{ij} + \epsilon}, \tag{7}$$

where ϵ is a small value added to prevent division by zero. We then compute the predicted probabilities by applying the softmax function to q:

$$m = \frac{e^{q}}{\sum_{j=1}^{k} e^{q_{ij}}}. \tag{8}$$

3.1.4 Why Mahalanobis distance in q_{ij}? Mahalanobis distance outweighs Euclidean distance when calculating the soft assignments q_{ij}, as Mahalanobis distance measures both the distance and correlation between the latent space representations z_i and cluster centers c_j, while Euclidean distance only measures the magnitude between two features. In tabular data with dense features, where many dimensions show strong correlations, Mahalanobis distance captures these through the covariance matrix Σ, and q_{ij} maintains the proximity of z_i to c_j along with the feature dependencies. The positive covariance matrix Σ shows that dense features with overlapping semantics are highly weighted in q_{ij}. For example, columns *Location*, *City* and *Country* share overlapping semantics but differ in representation. Mahalanobis distance calculates these correlations in q_{ij} and assigns high similarity scores to these features.

3.2 Loss Function

In this section, we define the loss function to produce a low-dimensional representation that captures the original embeddings' key features and gives rise to suitable clusters (as illustrated in Figure 1). Since m acts as clustering probabilities and can be further optimized, we integrate KL-divergence to refine the clustering probabilities m to be more accurate and representative of the underlying data. KL-divergence provides a flexible way to guide the model toward the expected clustering assignments. We use the following objective function to optimize the m.

$$ce_{loss} = \mathrm{KL}_{\mathrm{div}}(p||m) = \sum_{i=1}^{n}\sum_{j=1}^{k} p_{ij} \cdot \log \frac{p_{ij}}{m_{ij}}, \tag{9}$$

$$p_{ij} = \frac{q_{ij}^2}{\sum_{i=1}^{n} q_{ij}}, \tag{10}$$

ce_{loss} is clustering loss, where p_{ij} is the target distribution, calculated by adjusting the soft assignments q. This step aims to make the soft assignments sharper, thereby improving the stability

Algorithm 1 Pseudo-code for training of TableDC

Require: x: representation of rows, columns or tables; $\mathbb{K}$: number of clusters; $maxiter$: maximum number of iterations
Ensure: m: clustering probabilities
Initialize W_e, b_e, W_d, b_d by pre-training autoencoder
Initialize cluster centers c using *Birch* algorithm
for $i = 1$ to length($maxiter$) **do**
  Generate latent representation z from autoencoder using (1)
  Compute inverse of covariance matrix Σ^{-1} using Cholesky decomposition from (5)
  Calculate Mahalanobis distance D_{M} between z and c using (3)
  Apply Cauchy distribution on D_{M} to get soft assignments q from (6)
  Normalize q and apply softmax to get predicted probabilities m
  Calculate target distribution p using (10)
  Calculate ce_{loss} and re_{loss} from (9) and (11) respectively
  Back propagate and update parameters in TableDC
end for
return Updated m

and discriminative control of the clustering. We use the following reconstruction loss (re_{loss}) to maintain the inherent structure and essential features in a compressed form.

$$re_{loss} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x}_i)^2, \tag{11}$$

where $\bar{x}$ is the reconstructed input. ce_{loss} and re_{loss} work together to achieve effective clustering while maintaining the quality of the data reconstruction. The total loss is given by:

$$loss = \alpha \cdot ce_{loss} + re_{loss}, \tag{12}$$

where α is a weighting factor that balances the contribution of the ce_{loss} and the reconstruction loss (re_{loss}) to the total loss. Combining ce_{loss} and re_{loss} enables autoencoder learning to generate meaningful latent representations that can be effectively clustered while preserving the structure and information in the input data. The choice of the hyperparameter α allows for fine-tuning the trade-off between clustering performance and reconstruction quality. We use $\alpha = 0.9$ (Equation 12), so TableDC gives more importance to the clustering loss, ce_{loss} (Equation 9) than re_{loss}. We determined the weight of $\alpha = 0.9$ for the ce_{loss} through a grid search.

In the clustering loss, ce_{loss} (Equation 9), m is a softmax-ed q consistently diverging from the target distribution p (see ablation study in Section 5.2). After applying softmax, q becomes a probability distribution m over clusters. Each row of the probability matrix has a sum of 1, representing the probability for each point to belong to each cluster. The softmax function emphasizes the significant values in a probability matrix m while suppressing smaller values, making TableDC soft assignments sharper.

The presence of noise or identical probabilities towards more than one cluster provokes the model to make confident cluster assignments despite inherent ambiguities. p is derived from q by re-normalizing to increase high-probability assignments and reduce low-probability ones. The high divergence of m from the sharpened p indicates that TableDC becomes more confident, with q growing sharper over epochs. Unlike p, which prioritizes intra-cluster separation, m develops to align better with the true data structure. p is an auxiliary target to guide clustering and is not the GT, so divergence from p in overlapping clusters is not inherently poor. Considering this

phenomenon, we can consider the KL-divergence as a maximization problem where we prefer m to have a maximum divergence from p and can be defined as a sequence of KL-divergences over N iterations: $\{D_{KL}(p \parallel m^1), D_{KL}(p \parallel m^2), \ldots, D_{KL}(p \parallel m^N)\}$ where $D_{KL}(p \parallel m^k)$ represents the KL-divergence between p and m at the k^{th} iteration. The KL divergence increases over N if: $D_{KL}(p \parallel m^{k+1}) > D_{KL}(p \parallel m^k)$ for all k in $\{1, 2, \ldots,$N-1$\}$.

The pseudo-code for training of TableDC is given in in Algorithm 1, showing how the different definitions are used together.

3.3 Initialisation

In the training phase of DC algorithms, the quality of the initial centroids can directly affect the quality of the final clusters. Poor initialization can lead to poorly formed clusters, such as unevenly sized, too small, or too large. Existing DC methods [6, 53, 80] use K-means to initialize cluster centers c. However, in data integration problems dealing with highly dense and overlapping embedding representation, K-means may not be a suitable initialization (see ablation study in Section 5) for the following reasons:

(1) In densely packed and overlapping data dimensions, the Euclidean distance between c and z becomes uniform, which leads to poor cluster initialization.
(2) Many sub-optimal solutions exist in a dense data space, and K-means are not optimal enough to avoid local minima.
(3) K-means initialization is based on hard assignment, where the centers are randomly initialized. However, dense data with overlapping clusters requires soft assignments based on probabilities.

K-means initialization is often considered suitable in existing DC proposals due to the nature of the data, which is mainly sparse or semi-dense, particularly in image representations. The Euclidean distance between different points becomes more distinct in a sparse data space, resulting in well-separated clusters. This distinct separation enhances the effectiveness of K-means initialization in these contexts, facilitating the clustering process by providing a clear starting point for the algorithm.

In TableDC, we integrate the Birch algorithm [95] to initialize the clusters. Birch is a traditional hierarchical clustering algorithm that handles large and noisy databases. Birch uses a Clustering Feature tree (CF-tree) to process each data point. A CF-tree is a set of nodes, and each node consists of a triplet that efficiently stores information on data points in a compressed format to the cluster instead of processing a complete dataset. The triplet includes the number of data points per cluster, squared, and linear sum. In a dense space, each node of the CF-tree represents aggregated properties of each data point instead of the distance between data points, which enables Birch initialization to avoid close proximity and high overlaps. Since Birch is hierarchical, the multiple levels of the CF-tree can effectively capture the different granularities of clusters even if the overlaps exist on one level. We compare different cluster center initialization techniques in the ablation study (see Section 5.1). The pseudo-code for the initialization of c using Birch is given in Algorithm 2).

4 Evaluation

This section evaluates TableDC using existing benchmarks with a variety of different embedding approaches, in comparison with other clustering algorithms and bespoke solutions. In all cases, TableDC and baselines carry out the minimum number of steps required to cluster an input dataset (consisting of tables, rows or columns) and produce as output a set of clusters. For the DC baselines, typically there are three steps in a pipeline: computation of embeddings, initialization using an SC,

Algorithm 2 Pseudo-code for initialization of c using Birch

Require: z: data points; K: number of clusters; T: threshold; B: branching factor (the maximum number of child nodes); L: The maximum number of entries in a leaf node
Ensure: c: cluster centers and cluster assignments for each z

```
Create an empty CF-tree with T, B, and L.
Initialize Birch clustering with K
for each z in the dataset do
    Traverse the CF-tree to find a suitable corresponding leaf node
    Update the CF-tree in the path from the root to the last leaf node
    if the value of leaf node > T then
        Split the leaf node
    end if
end for
Compute clusters k and assign a cluster index to each z
for each k do
    Compute c as a mean of z assigned to k
    Add c to the list of cluster centers
end for
return c_i = c
```

and then a single step that combines representation learning and clustering. For the SC baselines, there are typically two steps, computation of embeddings followed by clustering.

4.1 Experimental setup

4.1.1 Datasets. TableDC[1] is evaluated on seven existing benchmark datasets. Brief descriptions of each data set are given below and in Table 1, which also indicates which benchmarks are used for clustering each of tables, rows and columns.

T2D Entity-Level Gold standard (T2D) [73] is a widely used collection of web tables with GT mappings to their corresponding DBpedia class. For example, a table of *Persons* with columns *Name*, *Country* and *Year* and a table of *Companies* with columns *Location*, *Country* and *Year* share columns that are syntactically similar, such as *Country* and *Year*. However, these tables should not be clustered together, as they represent different concepts with distinct semantics.

Table Union Search (TUS) [60] is a benchmark of unionable tables. For clustering, each GT cluster has tables sharing unionable columns. For example, the attributes *hospital name* and *country* in one table and are likely to be considered to be unionable with the attributes *medical center* and *location* from another table. To define clustering boundaries for the TUS dataset, we constructed a graph where nodes represent tables and edges indicate pairs of tables with unionable columns. We use a community detection algorithm [5] to partition the graph into clusters of similar tables.

Music Brainz [46] is a dataset derived from authentic records of songs sourced from the Music Brainz database. It combines the data from five distinct sources, where approximately 50% of the original song records are duplicated across two to five sources.

Geographic Settlements (GeoSet) [46] represents a collection of geographical real-world entities from four different sources (Geonames, Freebase, DBpedia and NYTimes).

DBLP-Scholar [46] is a widely used entity resolution dataset of citation records from two bibliographic databases, DBLP and Google Scholar. While DBLP-Scholar provides GT pairs for

[1]https://github.com/hafizrauf/TableDC

Table 1. Statistics of benchmark datasets used to evaluate TableDC on tables, rows and columns.

	Dataset	Instances	Clusters
Tables	web tables	429	26
	TUS	4248	37
Rows	Music Brainz	2002	684
	GeoSet	3021	786
	DBLP-Scholar	5317	1164
Columns	Camera	19036	56
	Monitor	34481	81

evaluation, in clustering, we require GT clusters. To obtain GT clusters, we apply transitive closure on the GT pairs to form clusters. We consider only GT clusters containing at least 3 entities .

Di2KG (Camera and Monitor)[2] contains a set of camera and monitor web records scraped from multiple e-commerce web pages, and can be used for benchmarking domain discovery [66]. Both datasets are highly heterogeneous. For example, *lcd display* from *www.price-hunt.com* and *monitor* from *www.cambuy.com.au* represent the same domain despite having different syntactic structures.

4.1.2 Benchmark Methods. We compared TableDC with three SC and five DC clustering methods. The selected SC methods represent different paradigms within the clustering domain, ensuring a comprehensive evaluation across diverse clustering strategies. Similarly, the chosen DC methods were carefully selected to cover a range of approaches to representation learning, including both self-supervised and subspace learning techniques. We also considered their applicability to data management tasks, and those selected DC methods were identified as top performers.

A brief description of each benchmark method is provided below:

K-means [34] is a classical clustering algorithm; it assigns instances to the nearest centroid to create clusters. Centroids are updated based on the assigned points.

DBSCAN [26] identifies distinct clusters based on the data density, considering the number of neighboring points within a specified radius. DBSCAN is particularly good with data having variation in densities.

Birch [95] is a hierarchical clustering algorithm built on a CF-tree to summarize the information (on each tree node) needed to cluster instances.

SHGP [90] is a self-supervised method that shares an attention-aggregation mechanism between two modules, *Att-LPA* to produce pseudo-labels and *Att-HGNN*, to learn object embeddings. Both modules effectively improve each other to optimize embeddings.

DCRN [53] learns representations through AE and GCN by reducing the information correlation to improve the discriminative property of the resulting features.

DFCN [80] focuses on the fusion of representation learning of AE and graph neural networks. It works on a dynamic fusion mechanism to refine the graph neural network structure to better integrate with AE representation learning.

EDESC [9] is a subspace representation-based learning method that iteratively refines the bases of the subspace derived from deep representations.

SDCN [6] utilizes the structural information in the AE representation learning. The dual self-supervised mechanism effectively updates the model parameters to produce clustering-specific representations.

[2]http://di2kg.inf.uniroma3.it/datasets.html

4.1.3 Representations. Each data integration task has an associated embedding model for representing tables, rows or columns which is integrated into the DC method for clustering. We use six different embedding models, each of which is specific to one or more data integration tasks; details are given below:

Table Clustering: We use pre-trained embeddings to get the representations of raw data. We categorized these into two categories. When we consider headers, we use SBERT [72], FastText [32] and USE [13]; with instances, we use TabTransformer and SBERT. SBERT uses siamese and triplet network structures to extract semantically meaningful embeddings from sentences. FastText is specialized in extracting morphological information and considers subwords, representing each word as a bag of character n-grams. Like SBERT, USE is also a sentence encoder specializing in capturing the semantic meanings of sentences. Considering the instances along with the header is a more tricky task and requires more robust transformers. We fine-tuned TabTransformer [36] on both datasets to benefit the categorical and contentious features in the dataset. We also encoded instance data with SBERT; we convert each row of the table into a sentence sequence, where each row is represented as a sequence of its cell values appended with *[SEP]* token for SBERT to separate segments and maintain the row boundaries in the text. *[SEP]* carries the table structure within the BERT model in the encoded form.

Row Clustering: Alongside SBERT and USE, we train EmbDi [10] to embed rows. EmbDi is a graph-based representation method based on tripartite connections that contain a column node, a value node and a row node.

Column Clustering: Similar to table clustering, we embed columns with and without column headers. We used SBERT, T5 [68] and USE [13] pre-trained embeddings for both purposes. When considering instances, we embed column values using SBERT, T5 and USE and aggregate the individual embeddings to get the final column embeddings. T5 is a widely used language model developed by Google for information retrieval tasks and generating embeddings, specifically capturing semantic information within data. As EmbDi provides column embeddings, we also use EmbDi for domain discovery; the additional results are provided in the GitHub [3].

4.1.4 Metrics. We adopted two widely used standard clustering evaluation metrics, Accuracy (ACC) [91] and Adjusted Rand Score (ARI) [83] to test the performance of TableDC and existing benchmark methods. Unlike the Rand Index (RI) [69], which evaluates clusters without correcting for chance pairs, the ARI corrects for chance pairs by normalizing the RI with respect to its expected value under random clustering assignments [37]. This correction is crucial in cases with a large number of clusters, where the majority of clear "non-matches" could otherwise distort the results. ARI ranges from -1 to 1, where 1 indicates perfect matching, 0 indicates random labeling and negative values indicate worse than random labeling. ACC ranges from 0 to 1.

4.1.5 Parameter setting. TableDC consists of four AE layers and is trained using the Adam optimizer. The loss function combines KL-divergence (to minimize the Mahalanobis distance in self-supervised learning) and mean squared error loss for reconstruction. The latent space size is fixed at 100. In order to be consistent with the existing DC benchmarks, we pre-trained TableDC with AE on 30 epochs for table and column clustering. Pretraining being unsupervised effectively initializes network parameters, avoiding random initialization and local minima during the clustering phase, leading to faster convergence. Since entity resolution involves many clusters, the CF-tree in Birch initialization needs more nodes with detailed cluster summaries, each representing a smaller and more defined cluster. We have pre-trained TableDC on 100 epochs for entity resolution. Training epochs for table clustering are fixed to 200, column clustering to 100, and row clustering to 50. We

[3]https://github.com/hafizrauf/TableDC

run all existing benchmark methods on the same number of epochs adopted for TableDC, excluding the internal parameters for which we use the originally published values for each existing method, such as the number and size of each layer. We tuned the threshold T (the maximum range of radius of a cluster in the CF-tree) when initiating the centroids in TableDC for those experiments where the number of subclusters found on each CF-tree node is less to be able to summarize cluster information. We adopted a grid search on threshold T and divided the number of sub-clusters unless the CF-tree becomes stable. Some existing methods need K-means for centroids initialization and clustering assignments. To avoid extreme cases, we initialize 20 times and choose the best solution.

4.1.6 Role of $\mathbb{K}$. It is common for clustering algorithms to take as input a target number of clusters $\mathbb{K}$, which is an issue for applications in which $\mathbb{K}$ is not known *a priori*. Where this is the case, clustering is often a two step process: *(i)* estimating $\mathbb{K}$ (several methods are used and compared in [77]) — let us call the input estimate $\mathbb{K}_i$; *(ii)* passing $\mathbb{K}_i$ into the clustering algorithm as a parameter.

In DC algorithms, including TableDC, $\mathbb{K}_i$ is just a hint used in initialization, and thus the algorithms can produce an output number of clusters $\mathbb{K}_o$ such that $\mathbb{K}_i \neq K_o$. During training, TableDC and baselines dynamically adjust the clusters and produce a different output $\mathbb{K}_o$ (number of clusters) from the input $\mathbb{K}_i$. To ensure a fair comparison, all baseline methods compared in our experiments are also provided with $\mathbb{K}$ or equivalent parameters that influence their clustering outcomes. We use the GT $\mathbb{K}$ during evaluation to isolate the core clustering mechanisms and focus on their performance rather than their reliance on $\mathbb{K}$. This setup emphasizes that providing $\mathbb{K}$ in TableDC and the baselines is not a limitation but a practical design choice. Below, we summarize where the $\mathbb{K}_i$ and similar parameters are used in the baseline methods.

- **K-means:** $\mathbb{K}_i$ is explicitly provided as an initialization hint for centroids and directly defining the clustering structure.
- **DBSCAN:** DBSCAN does not use $\mathbb{K}_i$. Instead, it relies on parameters `eps` (maximum distance between points) and `min_samples` (minimum points to form a dense region), which indirectly influence $\mathbb{K}_o$.
- **Birch:** $\mathbb{K}_i$ is provided to finalize the number of clusters after constructing the initial clustering hierarchy.
- **SHGP:** $\mathbb{K}_i$ is used to initialize pseudo-labels and guide the grouping process in the graph-based representation.
- **DCRN:** $\mathbb{K}_i$ is given to initialize the clustering assignment matrix, which is further refined in the clustering process.
- **DFCN:** $\mathbb{K}_i$ is given to initialize cluster centers and guide both node-level and graph-level fusion processes.
- **EDESC:** $\mathbb{K}_i$ is specified as a hint for initializing cluster centroids and guiding the subspace clustering process.
- **SDCN:** $\mathbb{K}_i$ is used to initialize cluster centers via K-means as a starting point for clustering assignment.
- **TableDC:** $\mathbb{K}_i$ is used to derive the initial number of clusters.

4.2 Schema inference clustering results

Schema inference as a clustering problem can be defined as the clustering of tables that share a similar schema. Table 2 shows the clustering results of clustering tables from the TUS and web tables datasets encoded with several representations.

We observe the following: (i) *TableDC obtained the best results over both datasets when considering schema-level data* (SBERT, FastText, USE) compared to all benchmark methods, including SC. TableDC improves the clustering performance (with SBERT) by 0.28 and 0.27 ARI and 0.30 and 0.15

Table 2. Comparing schema inference clustering results (TableDC vs. existing DC methods). The best results are shown in bold. Representations marked with ∗ are obtained on instance-level data and TT^* refers to TabTransformer.

Method	TUS								Web tables							
	SBERT		FastText		TT^*		$SBERT^*$		SBERT		USE		TT^*		$SBERT^*$	
	ARI	ACC	ARI	ACC	ARI	ACC	ARI	ACC	ARI	ACC	ARI	ACC	ARI	ACC	ARI	ACC
K-means	0.73	0.79	0.63	0.69	0.22	0.35	0.46	0.56	0.27	0.45	0.19	0.40	-0.013	0.27	0.32	0.51
DBSCAN	0.17	0.47	0.01	0.25	0.02	0.26	0.09	0.38	0.0	0.29	0.0	0.29	-0.007	0.26	0.0	0.29
Birch	0.22	0.40	0.0	0.20	0.18	0.35	0.54	0.58	0.33	0.49	0.22	0.40	-0.013	0.27	0.46	0.61
SHGP	0.61	0.72	0.43	0.56	0.05	0.19	0.13	0.29	0.08	0.29	0.07	0.28	-0.0031	0.27	0.09	0.30
DCRN	0.50	0.59	∞	∞	0.018	0.24	0.15	0.26	0.07	0.25	0.03	0.25	-0.02	0.25	0.17	0.36
DFCN	0.55	0.65	0.43	0.52	0.20	0.37	0.34	0.43	0.24	0.40	0.19	0.33	-0.0006	0.24	0.37	0.48
EDESC	0.28	0.35	0.22	0.32	0.14	0.30	0.36	0.43	0.26	0.44	0.22	0.37	**0.08**	0.25	-0.04	0.26
SDCN	0.60	0.57	0.53	0.47	**0.25**	**0.40**	0.45	0.49	0.20	0.34	0.08	0.33	0.0	**0.29**	0.30	0.42
TableDC	**0.88**	**0.87**	**0.73**	**0.76**	0.24	0.37	**0.63**	**0.66**	**0.62**	**0.65**	**0.25**	**0.42**	-0.006	0.26	**0.61**	**0.65**

ACC, compared to SDCN and SHGP, respectively. (ii) *TableDC exhibits good but not always best performance on instance-level data* (TabTransformer, SBERT) and was outperformed by SDCN by a small margin (0.01 ARI and 0.03 ACC) on the TUS dataset and by EDESC on the web tables dataset using TabTransformer. However, TableDC outperformed the other methods with SBERT, improving clustering performance by 0.31 average ARI and 0.23 average ACC on the TUS dataset. (iii) *TableDC handles overlapping clusters due to its heavy-tailed Cauchy distribution as a similarity kernel.* For example, TableDC correctly assigns two tables that describe different concepts but share similar schemas, *RadioStation(Radio, Country, Language)* and *Country(Country, Language, Broadcasters)* to two different clusters, in contrast with SDCN and DFCN, which put them in one cluster. (iv) *TableDC provides effective distance measures in a dense space through the Mahalanobis distance measure when calculating the distance between data points.* Two instances from different GT clusters are closely packed in the latent space when the distance measured is Euclidean compared to the Mahalanobis distance, which helps the model by placing both instances apart. For example, the Mahalanobis distance between different attributes of two tables *Book(rank, title, genre, creator, date, freq)* and *Film(rank, title, year, director, overall rank)* is 0.99 compared to the Euclidean distance, which is 0.71. Experimentally, TableDC correctly placed the two instances in different clusters whereas EDESC and DCRN incorrectly placed them into one cluster due to the low Euclidean distance.

4.3 Entity resolution clustering results

Entity resolution involves clustering multiple records that refer to the same real-world entity. Our work addresses multi-source entity resolution, in that is can cluster data from heterogeneous datasets. Table 3 shows the clustering results of clustering rows. Experiments that did not scale on available resources are reported as *N/A*

We observe the following: (i) *The best results are with TableDC on Music Brainz and GeoSet*, outperforming all other methods by 0.63 and 0.47 average ARI on Music Brainz and 0.31 and 0.21 average ARI on GeoSet using SBERT and USE, respectively. (ii) *TableDC performed well with SBERT compared to EmbDi,* delivering good semantic similarity performance among different records of the same real-world entity. TableDC is better with SBERT than EmbDi by 0.20 ARI with Music Brainz, and 0.05 ARI with GeoSet. (iii) *TableDC with EmbDi on GeoSet created fewer unary clusters (50) than SHGP (101) and EDESC (90)*, indicating that TableDC effectively balances cluster purity and cluster size. A higher ARI score with fewer unary clusters implies that TableDC is better at avoiding unnecessary fragmentation of the rows into too many small clusters. (iv) *TableDC outperformed all baselines for DBLP-Scholar and obtained an average improvement of 0.19 ARI with SBERT and 0.05 ARI with EmbDi.* As DBLP-Scholar is characterized by a large number of small-sized clusters,

Table 3. Comparing entity resolution clustering results (TableDC vs. existing DC methods). The best results are shown in bold.

	Music Brainz						**GeoSet**						**DBLP-Scholar**					
Method	SBERT		EmbDi		USE		SBERT		EmbDi		USE		SBERT		EmbDi		USE	
	ARI	ACC	ARI	ACC	ARI	ACC	ARI	ACC	ARI	ACC	ARI	ACC	ARI	ACC	ARI	ACC	ARI	ACC
K-means	0.40	0.68	0.39	0.64	0.45	0.70	0.57	0.83	0.56	0.71	0.53	0.75	0.21	0.62	0.06	0.37	0.15	0.53
DBSCAN	0.0	0.002	N/A	N/A	0.0	0.002	0.0	0.001	0.0	0.001	0.0	0.01	-0.00	0.04	-0.00	0.04	N/A	N/A
Birch	0.56	0.76	0.41	0.67	0.55	0.76	0.56	0.75	0.59	0.71	0.50	0.70	0.25	0.66	0.09	0.44	0.17	0.55
SHGP	0.15	0.47	0.20	0.51	0.32	0.61	0.52	0.77	0.43	0.63	0.51	0.72	0.24	0.66	0.04	0.30	0.18	0.59
DFCN	0.03	0.31	0.03	0.33	0.05	0.35	0.34	0.59	0.37	0.57	0.26	0.50	0.11	0.38	0.05	0.30	N/A	N/A
EDESC	-0.001	0.03	0.03	0.29	0.05	0.33	0.30	0.73	0.25	0.52	0.38	0.63	0.17	0.53	0.03	0.25	0.09	0.42
SDCN	0.002	0.13	0.06	0.47	0.005	0.25	0.05	0.66	0.52	0.66	0.28	0.64	0.08	0.40	0.03	0.28	0.01	0.18
TableDC	**0.80**	**0.88**	**0.51**	**0.71**	**0.68**	**0.81**	**0.65**	**0.86**	**0.60**	0.71	**0.57**	**0.78**	**0.35**	**0.82**	**0.10**	0.44	**0.28**	**0.72**

TableDC effectively handles this structure. (v) *Unlike TableDC, existing clustering methods fail to consider the context of specific entities.* For instance, SDCN and SHGP with EmbDi incorrectly cluster two distinct songs—*(title: Jabberwock, length: 292706, year: 2010, language: French)* and *(title: Bubble Star, length: 244346, year: 2004, language: French)*—due to shared attributes like *language* and similar numeric ranges, despite their differing *titles*.

4.4 Domain discovery clustering results

Domain discovery can be defined as a clustering of the columns from a collection of datasets that refer to the same domain. Table 4 shows the result of clustering columns from Camera and Monitor datasets encoded with several representations.

We observe the following: (i) *TableDC obtained the best results with USE, SBERT and T5 when considering the Camera dataset*, outperforming baseline standard and deep clustering methods by an average of 0.23 ARI with USE, 0.32 ARI with SBERT and 0.30 ARU with T5. (ii) *TableDC also led on the Monitor dataset* by an average of 0.20 ARI with USE, 0.18 ARI with SBERT, and 0.14 with T5. (iii) *TableDC shows a superior capacity for interpreting the column's context in the latent space compared to other methods.* For example, in the Camera dataset, the cosine similarity of the T5 vectors of two columns *(image sensor: CMOS)* and *(optical sensor: CMOS)* is 0.61, and TableDC learns the context and assigns both columns to the same cluster. This is in contrast with SDCN and DFCN, which incorrectly assign them to different clusters. (iv) *The clustering performance of existing methods is affected by the length of columns, in contrast with TableDC, which handles the columns uniformly regardless of the length.* For example, two columns of the Camera dataset with different lengths *(camera color: silver gray/ black/ red/ silver)* and *(color: black)* are categorized in different clusters by DCRN with SBERT; however, TableDC overrides the column structure and correctly puts them together in the same cluster.

4.5 Comparison with bespoke solutions

4.5.1 Schema Inference. Although there are many proposals for schema inference techniques, as surveyed in [12, 42], few of these cluster tables. Thus the comparators in this section use state-of-the-art techniques for table similarity along with standard clustering algorithms. Specifically for table similarity, we use: (i) D^3L [7], which combines the results of several largely syntactic methods for comparing column headers and values into an overall table similarity score. D^3L is used with K-means for clustering because it exhibits the best results compared to Birch and DBSCAN. (ii) Starmie [28], which uses self-supervised contrastive learning to fine tune ROBERTa [52] language models for column similarity; table similarity is then supported by combining these column similarities. Starmie is used with the connected component algorithm [28] as in the original paper.

Table 4. Comparing domain discovery clustering results (TableDC vs. existing DC methods). The best results are shown in bold.

Method	Camera								Monitor							
	SBERT		*USE**		*SBERT**		*T5**		SBERT		*USE**		*SBERT**		*T5**	
	ARI	ACC	ARI	ACC	ARI	ACC	ARI	ACC	ARI	ACC	ARI	ACC	ARI	ACC	ARI	ACC
K-means	0.74	0.70	0.60	0.62	0.49	0.56	0.49	0.56	0.59	0.58	0.50	0.51	0.54	0.53	0.52	0.53
DBSCAN	0.73	0.69	0.0	0.12	-0.005	0.25	-0.005	0.25	0.27	0.50	0.0	0.06	0.007	0.22	0.006	0.22
Birch	0.76	0.70	0.71	0.69	0.78	0.74	0.69	0.68	0.55	0.55	0.56	0.55	0.60	0.61	0.53	0.53
SHGP	0.65	0.65	0.47	0.52	0.47	0.56	0.41	0.51	0.59	0.60	0.44	0.47	0.48	0.52	0.46	0.49
DCRN	0.60	0.60	N/A	N/A	0.41	0.44	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
DFCN	0.77	0.72	0.59	0.58	0.64	0.68	0.57	0.61	0.59	0.59	N/A	N/A	0.53	0.54	0.50	0.50
EDESC	0.41	0.48	0.64	0.60	0.57	0.59	0.46	0.54	0.32	0.42	0.42	0.44	0.48	0.50	0.46	0.49
SDCN	0.68	0.67	0.68	0.64	0.68	0.67	0.66	0.63	0.47	0.52	0.49	0.50	0.52	0.54	0.55	0.54
TableDC	**0.80**	**0.72**	**0.76**	**0.72**	**0.82**	**0.75**	**0.77**	**0.74**	**0.64**	**0.65**	**0.61**	**0.59**	**0.63**	**0.61**	**0.58**	**0.57**

4.5.2 Entity Resolution. There are many proposals for entity resolution techniques, as surveyed in [16, 24], though many of these consider pairwise similarity rather than producing clusters. Thus we include comparisons with a framework that can produce clusters of rows, namely JedAI [57, 63], which provides several workflows for entity resolution. Within JedAI, we use the schema agnostic workflow with cosine similarity, as TableDC is also schema agnostic; in essence, both techniques can be used to look for duplicates in tables with different structures. We considered two other unsupervised baselines to compare with TableDC: pre-trained models [93] and ZeroER [84], which generate pairwise similarity instead of clusters. We converted the generated pairs into clusters using transitive closure. In [93], several pre-trained models were evaluated in an end-to-end ER pipeline in a schema-agnostic setting, generating matching pairs. We report results for the best-performing language models. For ZeroER [84], as runtimes are high without blocking, we used overlap blocking on the *"title"* and *"name"* attributes with an overlap size of 2 to retain pairs with at least two common words.

4.5.3 Domain Discovery. There are relatively few fully unsupervised proposals for Domain Discovery. Here we use one bespoke method, namely D4 [62], which identifies domains by considering overlaps between column instances, and as in Schema Inference we use Starmie [28] along with an SC algorithm. The comparison with Starmie involves comparing a language model that has been fine-tuned for similarity with a language model in TableDC that is fine-tuned for clustering.

4.5.4 Results. We compare proposals for each problem that use techniques published in top outlets (SIGMOD, ICDE, PVLDB, Information Systems) since 2020. The results of the comparison of TableDC with bespoke solutions are presented in Figure 2 and Table 5, which reports the ARI and ACC for two datasets for each of *schema inference*, *entity resolution* and *domain discovery*.

We can observe that, for Schema inference, TableDC consistently outperforms D^3L and Starmie for web tables. For TUS, D^3L obtains 0.06 higher ACC than TableDC, while TableDC is still ahead with 0.07 ARI. This indicates that TableDC is good at identifying the overall clustering structure, but D^3L is better at assigning the exact labels correctly.

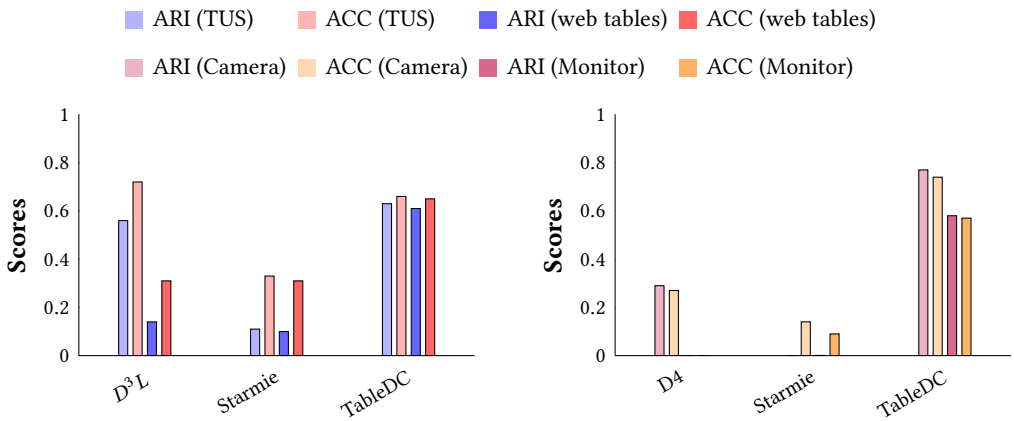
TableDC achieves good results in entity resolution, outperforming all three baseline ER pipelines. After TableDC, ZeroER and JedAI performed better than the pre-trained ER pipelines on the MusicBrainz dataset, achieving ARI scores of 0.60 and 0.58 and ACC scores of 0.75 and 0.69, respectively. However, on the GeoSet dataset, the pre-trained models, particularly S-GTR-T5 and S-MiniLM, outperformed both ZeroER and JedAI in terms of ARI.

In domain discovery, TableDC outperformed D4 (0.29 ARI and 0.27 ACC) in column clustering due to its ability to learn contextualized column embeddings effectively, providing more accurate clustering, with 0.77 ARI and 0.74 ACC on the Camera dataset.

Table 5. TableDC vs. bespoke solutions on entity resolution.

Methods	GeoSet		Music Brainz	
	ARI	ACC	ARI	ACC
JedAI (Cosine) [63]	0.32	0.64	0.58	0.69
DistilBERT	0.26	0.47	0.24	0.49
S-MPNet	0.37	0.51	0.37	0.58
S-GTR-T5	0.39	0.51	0.45	0.64
S-DistilRoBERTa	0.37	0.51	0.38	0.59
S-MiniLM	0.39	0.51	0.44	0.63
Word2Vec	0.25	0.49	0.33	0.56
FastText	0.25	0.49	0.17	0.45
ZeroER	0.28	0.61	0.60	0.75
TableDC	**0.65**	**0.86**	**0.80**	**0.88**

Overall, we can see that in all but one experiments, TableDC provides the best results, thereby demonstrating that, though generic, it compares well with recent, specialized, state-of-the-art proposals.



(a) **Schema Inference** (b) **Domain Discovery**

Fig. 2. TableDC vs. bespoke solutions. TableDC is integrated with SBERT in (a) and T5 in (b).

4.6 Scalability

We observe that existing DC proposals struggled to scale to large numbers of clusters $\mathbb{K}$ (see Figure 3). To compare the scalability of different DC methods, we used Music Brainz, which was scaled to large numbers of clusters, up to $\mathbb{K} = 2400$. The times are reported running on Nvidia A100 GPU with 80GB GPU RAM. We also reported run times for each data integration problem against each dataset, with the most instances provided in Table 6.

Most existing DC algorithms use GCN to create an adjacency matrix to capture underlying relationships among different features. This leads to a sparse adjacency matrix in scenarios where the representation is not dense. The complexity of GCN-based representation learning, assuming a

Table 6. Run time comparison of TableDC with existing DC benchmarks for schema inference, entity resolution, and domain discovery on each dataset with the most tables, rows, and columns, respectively.

	SI		ER		DD	
	*SBERT**	TT*	SBERT	EMBDI	*SBERT**	*T5**
DCRN	1870	1897.96	-	-	-	-
DFCN	876.80	1206.54	945.92	6192.95	33954	11850.95
EDESC	225.89	274.58	424.21	584.90	1327.3	1051.60
SDCN	591.34	1059.23	-	537.79	1975.42	-
TableDC	**129.85**	**173.41**	**392.01**	**281.67**	**749.63**	-

sparse adjacency matrix, is proportional to the number of vertices and edges. We often deal with dense representations with overlapping distances in data management applications. For dense representations, the multiplication between an adjacency matrix $\mathbb{A}$ (size $\mathbb{N} \times \mathbb{N}$) and a feature matrix (size $\mathbb{N} \times \mathbb{D}$) becomes more computationally intensive as $\mathbb{K}$ and $\mathbb{D}$ increase, leading to greater complexity.

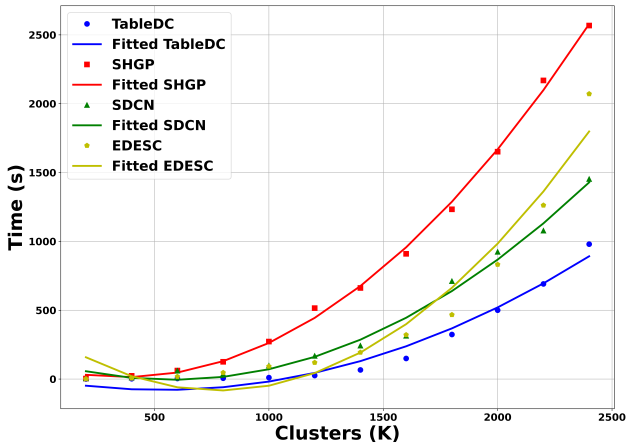


Fig. 3. Scalability comparison of TableDC with existing DC approaches with respect to the number of clusters $\mathbb{K}$. DFCN and DCRN are not included in the comparison because we have not managed to run both on the available hardware resources with a high number of clusters.

In contrast, TableDC does not rely on GCN to capture relationships but instead uses Mahalanobis distance to determine correlations between different features. TableDC employs an autoencoder with a linear relationship between the number of data points $\mathbb{N}$ and the size of the hidden layer $\mathbb{H}$. The self-supervised module, which calculates the differences between cluster centroids from dense representations, maintains a manageable computational load because the calculation is around the distance from each data point to every cluster centroid. TableDC's efficiency is optimized using Cholesky decomposition to compute the Mahalanobis distance. This method efficiently handles dense representations by decomposing the covariance matrix, allowing for faster computation of distances [31]. Additionally, the integration of a Cauchy distribution kernel normalizes these distances and applies softmax operations, ensuring that the computational overhead remains low even as the number of clusters $\mathbb{K}$ increases.

Table 7. Time complexity comparison of TableDC with existing DC methods with respect to a number of clusters $\mathbb{K}$. In EDESC, **M'** is the time complexity of total loss, which is $O(d^2 \times \mathbf{K})$ and increases with **K**, the time complexity of total loss grows faster than linear. *d* represents data dimensions.

DC method	Time complexity	
	With sparse $\mathbb{A}$	With dense $\mathbb{A}$
EDESC [9]	$O(\mathbb{N} \times \mathbf{M'})$	$O(\mathbb{N} \times \mathbf{M'})$
DCRN [53]	$O(\mathbb{N} \times \mathbb{K})$	$O(\mathbb{N}^2 \times \mathbb{K})$
SHGP [90]	$O(\lvert\mathbb{V}\rvert + \lvert\mathbb{E}\rvert)$	$O(\lvert\mathbb{V}^2\rvert + \lvert\mathbb{E}\rvert)$
DFCN [80]	$O(\mathbb{N} \times \mathbb{K} + NlogN)$	$O(\mathbb{N}^2 \times \mathbb{K} + NlogN)$
SDCN [6]	$O(\mathbb{N} \times \mathbb{K} + NlogN)$	$O(\mathbb{N}^2 \times \mathbb{K} + NlogN)$
TableDC	$O(\mathbb{N} + \mathbb{K}\log\mathbb{K})$	$O(\mathbb{N} + \mathbb{K}\log\mathbb{K})$

Let us analyze the time complexity of TableDC, which does not employ GCNs to capture relationships but instead uses the Mahalanobis distance to establish correlations between different features. TableDC utilizes an autoencoder with a time complexity of $O(\mathbb{N} \times \mathbb{H})$, where $\mathbb{H}$ is the size of the hidden layer. In the self-supervised module, the calculation of the difference between the cluster centroids and the dense representations results in a complexity of $O(\mathbb{N} + \mathbb{K}\log\mathbb{K} \times \mathbb{D})$. Since $\mathbb{D}$ (the latent space dimension) is constant, this simplifies to $O(\mathbb{N} + \mathbb{K}\log\mathbb{K})$. During matrix multiplications, Cholesky decomposition, and subsequent summations, the complexity remains $O(\mathbb{N} + \mathbb{K}\log\mathbb{K})$, as the primary calculations involve the distance from each data point to every cluster centroid. The $\mathbb{K}\log\mathbb{K}$ term reflects the cost of operations related to sorting or updating the cluster centroids, which are necessary but less frequent than the distance calculations in the self-supervised module. Additionally, BIRCH is used to initialize the cluster centroids. BIRCH clustering has a one-time complexity of $O(\mathbb{N} \times \log\mathbb{N})$ in the worst case. Since this initialization occurs only once during each run, it does not significantly affect the overall complexity. The dominant scaling factors are the number of data points $\mathbb{N}$ and the number of clusters $\mathbb{K}$. Since $\mathbb{K}$ is not significantly larger than $\mathbb{N}$, the overall time complexity can be described as quasi-linear, given by $O(\mathbb{N} + \mathbb{K}\log\mathbb{K})$. Table 7 presents a comparison of the time complexity of TableDC with existing deep clustering algorithms for both sparse and dense adjacency matrices.

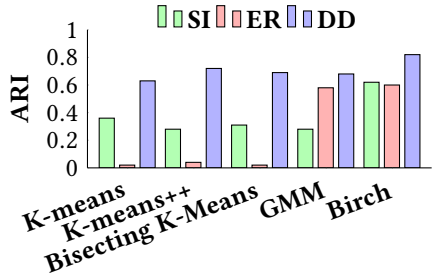
5 Ablation study

This section presents an ablation study to investigate the impact of three components: (i) cluster initialization, (ii) loss function, and (iii) different distance measures and similarity kernels on self-supervised learning.

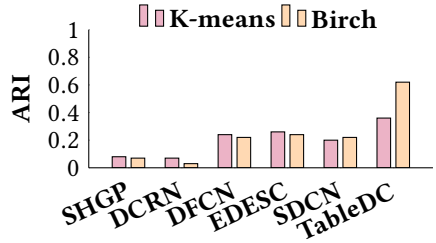
5.1 Cluster Initialization

In this ablation study, we conducted two experiments: (a) we compared several cluster centroid initialization approaches with Birch initialization for TableDC to investigate how cluster initialization affects clustering performance, and (b) we compared TableDC with existing baseline DC methods when replacing K-means with Birch to determine whether the baselines also gain a similar performance boost to that observed in TableDC. We recorded ARI using different initialization methods to evaluate cluster quality. Figure 4a presents a bar chart comparison of TableDC trained with different initialization methods, while Figure 4b compares Birch and K-means, as K-means has been widely used in existing DC algorithms.

We can observe from Figure 4a that TableDC with Birch initialization provides the best results for all problems. The higher ARI score of TableDC with Birch initialization indicates the accurate



(a) Different cluster initializations for TableDC



(b) Baselines initialization comparison.

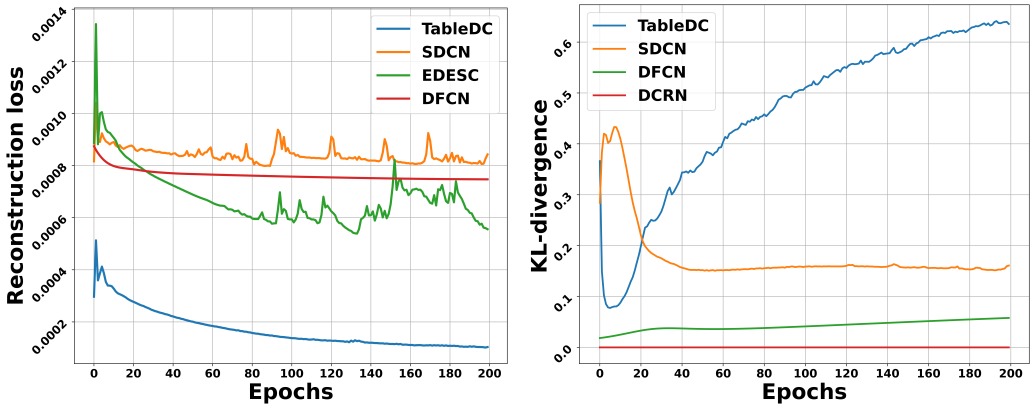
Fig. 4. This figure shows: (a) the impact of cluster initialization methods on clustering performance, using: SBERT on web tables for schema inference (SI), EmbDi on GeoSet for entity resolution (ER), and SBERT on Camera for domain discovery (DD). (b) the comparison of K-means and Birch for TableDC and baselines, evaluating if the baselines show a similar performance boost to TableDC when replacing K-means with Birch, using SBERT on web tables for SI.

identification of homogeneous clusters and clear separation between different clusters. K-means++ initialization appeared as the second best in domain discovery. From the comparative analysis, we also observed that with Birch initialization, the points are more widely dispersed around the centroid, indicating enhanced cluster compactness (how tightly the data points are grouped around the centroid of the cluster) and homogeneity (the similarity of data points within the same cluster).

We can see from Figure 4b that TableDC outperforms the baselines with both K-means and Birch initialization. Moreover, Birch initialization is not the best choice for the baselines, as most baselines experience a reduction in performance when K-means is replaced with Birch, except for SDCN, which shows a minimal boost of 0.02 ARI. Birch was explicitly designed for handling very large datasets with outliers, and in our setting, its initialization is tightly coupled with TableDC's self-supervised module to avoid close proximity and cluster overlaps. This property of Birch makes it more suitable for TableDC than the baselines.

5.2 Loss Function

To empirically assess the impact of loss functions in TableDC and existing benchmarks for data management problems, we use schema inference and web table data to plot *(i)* re_{loss} and *(ii)* $\mathrm{KL}_{\mathrm{div}}$ (see Figure 5). A well-trained AE with effective re_{loss} minimization, leads to embeddings where clusters are more distinct and separable, specifically in the cases of high density and overlap in the original embeddings. We can observe from Figure 5 that the re_{loss} of TableDC is significantly reduced and more consistent compared to the benchmark methods, which exhibit fluctuations in loss during the learning of dense embeddings. For $\mathrm{KL}_{\mathrm{div}}$, the relationship between p and q (see Section 3.2) is investigated to show if q deviates from p. This divergence is necessary because the data points in the target p require greater intra-cluster distance rather than high cohesion. We only consider those benchmarks using p and q distribution and $\mathrm{KL}_{\mathrm{div}}$ loss to align the comparison with TableDC. TableDC shows a high divergence (see Figure 5) of q from p, which means that TableDC is more confident with the actual soft assignments q becoming sharper over time, and over highly packed p, the q needs to minimize the cohesiveness.



(a) re_{loss} minimization

(b) divergence of q from p

Fig. 5. re_{loss} and $\mathrm{KL}_{\mathrm{div}}$ (between q and p) comparative analysis of TableDC with benchmark methods for schema inference on web tables data.

Table 8. TableDC clustering results on different distance metrics and similarity kernels used in the self-supervised module. We used SBERT (schema only) on web tables and Monitor datasets for schema inference and domain discovery. We adopted SBERT (values only) for entity resolution on the MusicBrainz dataset.

	Schema Inference			Entity Resolution			Domain Discovery		
Distance	Euclidean	Cosine	Mahalanobis	Euclidean	Cosine	Mahalanobis	Euclidean	Cosine	Mahalanobis
ARI	0.49	0.30	**0.62**	0.0014	0.0014	**0.88**	0.56	0.30	**0.64**
ACC	0.62	0.51	**0.65**	0.0599	0.0300	**0.87**	0.57	0.41	**0.65**
Distribution	Student's t	Normal	Cauchy	Student's t	Normal	Cauchy	Student's t	Normal	Cauchy
ARI	0.52	0.45	**0.62**	0.37	0.0	**0.88**	0.54	0.56	**0.64**
ACC	0.60	0.64	**0.65**	0.69	0.0025	**0.87**	0.06	0.0	**0.65**

5.3 Distance functions and similar kernels.

We evaluate the impact of difference distance measures and similarity kernels on the performance of TableDC. We record the ARI and ACC (see Table 8) of TableDC by keeping the Cauchy distribution as a similar kernel and replacing the Mahalanobis distance with Euclidean [6] and Cosine distances [76]. The Euclidean distance measures the straight-line distance between data points and their centroids, and the Cosine distance evaluates the cosine of the angle between vectors, emphasizing orientation over magnitude. Similarly, we kept the Mahalanobis distance while varying the similarity kernels. We replaced the Cauchy distribution with the Student's t-distribution [6] (with an additional degree of freedom parameter that adjusts the distribution's kurtosis.) and the Normal distribution (that provides a standard Gaussian decay of similarity).

From Table 8, we can observe that the choice of Mahalanobis distance combined with the Cauchy distribution appears optimal for all three problems. Euclidean distance with a Cauchy distribution performs relatively better in schema inference and domain discovery than cosine distance, showing that the straight-line distance between points is more reasonable than the angle when we have dense and overlapping clusters. With Student's t distribution, TableDC shows a high ARI for schema inference but significantly lower ACC than the Normal distribution. The higher ARI suggests that it can effectively group similar items into clusters, but the low ACC indicates that it failed to handle overlapping clusters and mismatches predicted labels against the actual labels.

6 Conclusions

This paper proposes a deep clustering algorithm for data cleaning and integration problems, and has demonstrated its applicability to applications that cluster tables, rows and columns. Our objective has been to improve the performance of deep clustering for data management tasks within the vast design space available for deep clustering. Our objective was not to produce an architecture that diverged substantially from earlier architectures—the novelty of the approach lies in its successful targeting of features of data management tasks, and it is effective for different problems.

TableDC utilizes the Mahalanobis distance to optimize data distribution. The strength of TableDC is its ability to account for the covariance among different dimensions of the embeddings. This ensures that the distance between data points remains consistent irrespective of the orientation of the data in the latent space. Our method enhances cluster separation in scenarios with overlapping features in the latent space by weighing dimensions based on the degree of their inter-dependencies. Experiments show that TableDC consistently outperforms existing clustering algorithms and problem-specific methods.

Acknowledgments

Hafiz Tayyab Rauf is supported by a studentship from The University of Manchester.

References

[1] Paul Albert, Eric Arazo, Noel E. O'Connor, and Kevin McGuinness. 2022. Embedding Contrastive Unsupervised Features to Cluster In- And Out-of-Distribution Noise in Corrupted Image Datasets. In *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXXI (Lecture Notes in Computer Science, Vol. 13691)*, Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (Eds.). Springer, 402–419. doi:10.1007/978-3-031-19821-2_23

[2] Humam Alwassel, Dhruv Mahajan, Bruno Korbar, Lorenzo Torresani, Bernard Ghanem, and Du Tran. 2020. Self-Supervised Learning by Cross-Modal Audio-Video Clustering. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.). https://proceedings.neurips.cc/paper/2020/hash/6f2268bd1d3d3ebaabb04d6b5d099425-Abstract.html

[3] Mohamed Amine Baazizi, Dario Colazzo, Giorgio Ghelli, and Carlo Sartiani. 2019. Parametric schema inference for massive JSON datasets. *VLDB J.* 28, 4 (2019), 497–521. doi:10.1007/s00778-018-0532-7

[4] Geert Jan Bex, Frank Neven, and Stijn Vansummeren. 2007. Inferring XML Schema Definitions from XML Data. In *Proc. 33rd VLDB*. 998–1009. http://www.vldb.org/conf/2007/papers/research/p998-bex.pdf

[5] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* 2008, 10 (2008), P10008.

[6] Deyu Bo, Xiao Wang, Chuan Shi, Meiqi Zhu, Emiao Lu, and Peng Cui. 2020. Structural Deep Clustering Network. In *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, Yennun Huang, Irwin King, Tie-Yan Liu, and Maarten van Steen (Eds.). ACM / IW3C2, 1400–1410. doi:10.1145/3366423.3380214

[7] Alex Bogatu, Alvaro A. A. Fernandes, Norman W. Paton, and Nikolaos Konstantinou. 2020. Dataset Discovery in Data Lakes. In *36th IEEE International Conference on Data Engineering, ICDE 2020, Dallas, TX, USA, April 20-24, 2020*. IEEE, 709–720. doi:10.1109/ICDE48307.2020.00067

[8] Angela Bonifati, Stefania Dumbrava, and Nicolas Mir. 2022. Hierarchical Clustering for Property Graph Schema Discovery. In *Proc. 25th EDBT*. 2:449–2:453. doi:10.48786/edbt.2022.39

[9] Jinyu Cai, Jicong Fan, Wenzhong Guo, Shiping Wang, Yunhe Zhang, and Zhao Zhang. 2022. Efficient Deep Embedded Subspace Clustering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*. IEEE, 21–30. doi:10.1109/CVPR52688.2022.00012

[10] Riccardo Cappuzzo, Paolo Papotti, and Saravanan Thirumuruganathan. 2020. Creating Embeddings of Heterogeneous Relational Datasets for Data Integration Tasks. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo (Eds.). ACM, 1335–1349. doi:10.1145/3318464.3389742

[11] Augustin Cauchy et al. 1847. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris* 25, 1847 (1847), 536–538.

[12] Sejla Cebiric, François Goasdoué, Haridimos Kondylakis, Dimitris Kotzinos, Ioana Manolescu, Georgia Troullinou, and Mussab Zneika. 2019. Summarizing semantic graphs: a survey. *VLDB J.* 28, 3 (2019), 295–327. doi:10.1007/s00778-018-0528-3

[13] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal Sentence Encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018*, Eduardo Blanco and Wei Lu (Eds.). Association for Computational Linguistics, 169–174. doi:10.18653/V1/D18-2029

[14] Mansheng Chen, Chang-Dong Wang, Dong Huang, Jian-Huang Lai, and Philip S. Yu. 2022. Efficient Orthogonal Multi-view Subspace Clustering. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, Aidong Zhang and Huzefa Rangwala (Eds.). ACM, 127–135. doi:10.1145/3534678.3539282

[15] Nenglun Chen, Lei Chu, Hao Pan, Yan Lu, and Wenping Wang. 2022. Self-Supervised Image Representation Learning with Geometric Set Consistency. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*. IEEE, 19270–19280. doi:10.1109/CVPR52688.2022.01869

[16] Vassilis Christophides, Vasilis Efthymiou, Themis Palpanas, George Papadakis, and Kostas Stefanidis. 2021. An Overview of End-to-End Entity Resolution for Big Data. *ACM Comput. Surv.* 53, 6 (2021), 127:1–127:42. doi:10.1145/3418896

[17] Adam Coates, Andrew Y. Ng, and Honglak Lee. 2011. An Analysis of Single-Layer Networks in Unsupervised Feature Learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011 (JMLR Proceedings, Vol. 15)*, Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudík (Eds.). JMLR.org, 215–223. http://proceedings.mlr.press/v15/coates11a/coates11a.pdf

[18] Gianni Costa, Giuseppe Manco, and Riccardo Ortale. 2010. An incremental clustering scheme for data de-duplication. *Data Min. Knowl. Discov.* 20, 1 (2010), 152–187. doi:10.1007/S10618-009-0155-0

[19] Chenhang Cui, Yazhou Ren, Jingyu Pu, Xiaorong Pu, and Lifang He. 2023. Deep Multi-view Subspace Clustering with Anchor Graph. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*. ijcai.org, 3577–3585. doi:10.24963/IJCAI.2023/398

[20] Nat Dilokthanakul, Pedro A. M. Mediano, Marta Garnelo, Matthew C. H. Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. 2016. Deep Unsupervised Clustering with Gaussian Mixture Variational Autoencoders. *CoRR* abs/1611.02648 (2016). arXiv:1611.02648 http://arxiv.org/abs/1611.02648

[21] Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. 2017. Deep Clustering via Joint Convolutional Autoencoder Embedding and Relative Entropy Minimization. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 5747–5756. doi:10.1109/ICCV.2017.612

[22] Uwe Draisbach, Peter Christen, and Felix Naumann. 2020. Transforming Pairwise Duplicates to Entity Clusters for High-quality Duplicate Detection. *ACM J. Data Inf. Qual.* 12, 1 (2020), 3:1–3:30. doi:10.1145/3352591

[23] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq R. Joty, Mourad Ouzzani, and Nan Tang. 2018. Distributed Representations of Tuples for Entity Resolution. *Proc. VLDB Endow.* 11, 11 (2018), 1454–1467. doi:10.14778/3236187.3236198

[24] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. 2007. Duplicate Record Detection: A Survey. *IEEE Trans. Knowl. Data Eng.* 19, 1 (2007), 1–16. doi:10.1109/TKDE.2007.250581

[25] Hossein Saeedi Emadi and Sayyed Majid Mazinani. 2018. A Novel Anomaly Detection Algorithm Using DBSCAN and SVM in Wireless Sensor Networks. *Wirel. Pers. Commun.* 98, 2 (2018), 2025–2035. doi:10.1007/s11277-017-4961-1

[26] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*, Evangelos Simoudis, Jiawei Han, and Usama M. Fayyad (Eds.). AAAI Press, 226–231. http://www.aaai.org/Library/KDD/1996/kdd96-037.php

[27] Fabian Falck, Haoting Zhang, Matthew Willetts, George Nicholson, Christopher Yau, and Chris C. Holmes. 2021. Multi-Facet Clustering Variational Autoencoders. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (Eds.). 8676–8690. https://proceedings.neurips.cc/paper/2021/hash/48cb136b65a69e8c2aa22913a0d91b2f-Abstract.html

[28] Grace Fan, Jin Wang, Yuliang Li, Dan Zhang, and Renée J. Miller. 2023. Semantics-aware Dataset Discovery from Data Lakes with Contextualized Column-based Representation Learning. *Proc. VLDB Endow.* 16, 7 (2023), 1726–1739. doi:10.14778/3587136.3587146

[29] Chakib Fettal, Lazhar Labiod, and Mohamed Nadif. 2023. Scalable Attributed-Graph Subspace Clustering. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023,*

Washington, DC, USA, February 7-14, 2023, Brian Williams, Yiling Chen, and Jennifer Neville (Eds.). AAAI Press, 7559–7567. doi:10.1609/AAAI.V37I6.25918

[30] Jeffrey Fisher, Peter Christen, Qing Wang, and Erhard Rahm. 2015. A Clustering-Based Framework to Control Block Sizes for Entity Resolution. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, Longbing Cao, Chengqi Zhang, Thorsten Joachims, Geoffrey I. Webb, Dragos D. Margineantu, and Graham Williams (Eds.). ACM, 279–288. doi:10.1145/2783258.2783396

[31] Gene Golub. 2013. *Matrix Computations.* Johns Hopkins University Press. doi:10.56021/9781421407944

[32] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomás Mikolov. 2018. Learning Word Vectors for 157 Languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation, LREC 2018, Miyazaki, Japan, May 7-12, 2018*, Nicoletta Calzolari et al. (Eds.). European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2018/summaries/627.html

[33] Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. 2017. Improved Deep Embedded Clustering with Local Structure Preservation. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, Carles Sierra (Ed.). ijcai.org, 1753–1759. doi:10.24963/IJCAI.2017/243

[34] J. A. Hartigan and M. A. Wong. 1979. Algorithm AS 136: A K-Means Clustering Algorithm. *Applied Statistics* 28, 1 (1979), 100. doi:10.2307/2346830

[35] Oktie Hassanzadeh, Fei Chiang, Renée J. Miller, and Hyun Chul Lee. 2009. Framework for Evaluating Clustering Algorithms in Duplicate Detection. *Proc. VLDB Endow.* 2, 1 (2009), 1282–1293. doi:10.14778/1687627.1687771

[36] Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar S. Karnin. 2020. TabTransformer: Tabular Data Modeling Using Contextual Embeddings. *CoRR* abs/2012.06678 (2020). arXiv:2012.06678 https://arxiv.org/abs/2012.06678

[37] Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification* 2 (1985), 193–218.

[38] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. 2017. Variational Deep Embedding: An Unsupervised and Generative Approach to Clustering. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, Carles Sierra (Ed.). ijcai.org, 1965–1972. doi:10.24963/IJCAI.2017/273

[39] Jiaqi Jin, Siwei Wang, Zhibin Dong, Xinwang Liu, and En Zhu. 2023. Deep Incomplete Multi-View Clustering with Cross-View Partial Sample and Prototype Alignment. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*. IEEE, 11600–11609. doi:10.1109/CVPR52729.2023.01116

[40] Turkay Kart, Wenjia Bai, Ben Glocker, and Daniel Rueckert. 2021. DeepMCAT: Large-Scale Deep Clustering for Medical Image Categorization. In *Deep Generative Models, and Data Augmentation, Labelling, and Imperfections - First Workshop, DGM4MICCAI 2021, and First Workshop, DALI 2021, Held in Conjunction with MICCAI 2021, Strasbourg, France, October 1, 2021, Proceedings (Lecture Notes in Computer Science, Vol. 13003)*, Sandy Engelhardt, Ilkay Öksüz, Dajiang Zhu, Yixuan Yuan, Anirban Mukhopadhyay, Nicholas Heller, Sharon Xiaolei Huang, Hien Van Nguyen, Raphael Sznitman, and Yuan Xue (Eds.). Springer, 259–267. doi:10.1007/978-3-030-88210-5_26

[41] Kenza Kellou-Menouer, Nikolaos Kardoulakis, Georgia Troullinou, Zoubida Kedad, Dimitris Plexousakis, and Haridimos Kondylakis. 2022. A survey on semantic schema discovery. *VLDB J.* 31, 4 (2022), 675–710. doi:10.1007/S00778-021-00717-X

[42] Kenza Kellou-Menouer, Nikolaos Kardoulakis, Georgia Troullinou, Zoubida Kedad, Dimitris Plexousakis, and Haridimos Kondylakis. 2022. A survey on semantic schema discovery. *The VLDB Journal* 31 (2022), 675–710.

[43] Kenza Kellou-Menouer and Zoubida Kedad. 2015. Schema Discovery in RDF Data Sources. In *Conceptual Modeling - 34th International Conference, ER (LNCS, Vol. 9381)*. Springer, 481–495. doi:10.1007/978-3-319-25264-3_36

[44] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[45] Yann LeCun, Ofer Matan, Bernhard E. Boser, John S. Denker, Don Henderson, Richard E. Howard, Wayne E. Hubbard, L. D. Jacket, and Henry S. Baird. 1990. Handwritten zip code recognition with multilayer networks. In *10th IAPR International Conference on Pattern Recognition, Conference C: image, speech, and signal processing, and Conference D: computer architecture for vision in pattern recognition, ICPR 1990, Atlantic City, NJ, USA, 16-21 June, 1990, Volume 2*. IEEE, 35–40. doi:10.1109/ICPR.1990.119325

[46] Database Group Leipzig. [n. d.]. Benchmark datasets for entity resolution. https://dbs.uni-leipzig.de/research/projects/benchmark-datasets-for-entity-resolution.

[47] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A New Benchmark Collection for Text Categorization Research. *J. Mach. Learn. Res.* 5 (2004), 361–397. http://jmlr.org/papers/volume5/lewis04a/lewis04a.pdf

[48] Keqian Li, Yeye He, and Kris Ganjam. 2017. Discovering Enterprise Concepts Using Spreadsheet Tables. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*. ACM, 1873–1882. doi:10.1145/3097983.3098102

[49] Xiaopeng Li, Zhourong Chen, Leonard K. M. Poon, and Nevin L. Zhang. 2019. Learning Latent Superstructures in Variational Autoencoders for Deep Multidimensional Clustering. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=

SJgNwi09Km
[50] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep Entity Matching with Pre-Trained Language Models. *Proc. VLDB Endow.* 14, 1 (2020), 50–60. doi:10.14778/3421424.3421431
[51] Fanzhen Liu, Shan Xue, Jia Wu, Chuan Zhou, Wenbin Hu, Cécile Paris, Surya Nepal, Jian Yang, and Philip S. Yu. 2020. Deep Learning for Community Detection: Progress, Challenges and Opportunities. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, Christian Bessiere (Ed.). ijcai.org, 4981–4987. doi:10.24963/ijcai.2020/693
[52] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* abs/1907.11692 (2019). arXiv:1907.11692 http://arxiv.org/abs/1907.11692
[53] Yue Liu, Wenxuan Tu, Sihang Zhou, Xinwang Liu, Linxuan Song, Xihong Yang, and En Zhu. 2022. Deep Graph Clustering via Dual Correlation Reduction. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*. AAAI Press, 7603–7611. doi:10.1609/AAAI.V36I7.20726
[54] Yue Liu, Xihong Yang, Sihang Zhou, Xinwang Liu, Zhen Wang, Ke Liang, Wenxuan Tu, Liang Li, Jingcan Duan, and Cancan Chen. 2023. Hard Sample Aware Network for Contrastive Deep Graph Clustering. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, Brian Williams, Yiling Chen, and Jennifer Neville (Eds.). AAAI Press, 8914–8922. doi:10.1609/AAAI.V37I7.26071
[55] Xin Ma and Won Hwa Kim. 2022. Locally Normalized Soft Contrastive Clustering for Compact Clusters. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, Luc De Raedt (Ed.). ijcai.org, 3313–3320. doi:10.24963/IJCAI.2022/460
[56] Prasanta Chandra Mahalanobis. 2018. On the generalized distance in statistics. *Sankhyā: The Indian Journal of Statistics, Series A (2008-)* 80 (2018), S1–S7.
[57] Georgios M. Mandilaras, George Papadakis, Luca Gagliardelli, Giovanni Simonini, Emmanouil Thanos, George Giannakopoulos, Sonia Bergamaschi, Themis Palpanas, Manolis Koubarakis, Alicia Lara-Clares, and Antonio Fariña. 2021. Reproducible experiments on Three-Dimensional Entity Resolution with JedAI. *Inf. Syst.* 102 (2021), 101830. doi:10.1016/J.IS.2021.101830
[58] Sudipto Mukherjee, Himanshu Asnani, Eugene Lin, and Sreeram Kannan. 2019. ClusterGAN: Latent Space Clustering in Generative Adversarial Networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*. AAAI Press, 4610–4617. doi:10.1609/AAAI.V33I01.33014610
[59] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, Johannes Fürnkranz and Thorsten Joachims (Eds.). Omnipress, 807–814. https://icml.cc/Conferences/2010/papers/432.pdf
[60] Fatemeh Nargesian, Erkang Zhu, Ken Q. Pu, and Renée J. Miller. 2018. Table Union Search on Open Data. *Proc. VLDB Endow.* 11, 7 (2018), 813–825. doi:10.14778/3192965.3192973
[61] Chuang Niu, Hongming Shan, and Ge Wang. 2022. SPICE: Semantic Pseudo-Labeling for Image Clustering. *IEEE Trans. Image Process.* 31 (2022), 7264–7278. doi:10.1109/TIP.2022.3221290
[62] Masayo Ota, Heiko Mueller, Juliana Freire, and Divesh Srivastava. 2020. Data-Driven Domain Discovery for Structured Datasets. *Proc. VLDB Endow.* 13, 7 (2020), 953–965. doi:10.14778/3384345.3384346
[63] George Papadakis, Georgios M. Mandilaras, Luca Gagliardelli, Giovanni Simonini, Emmanouil Thanos, George Giannakopoulos, Sonia Bergamaschi, Themis Palpanas, and Manolis Koubarakis. 2020. Three-dimensional Entity Resolution with JedAI. *Inf. Syst.* 93 (2020), 101565. doi:10.1016/J.IS.2020.101565
[64] Norman W. Paton, Jiaoyan Chen, and Zhenyu Wu. 2024. Dataset Discovery and Exploration: A Survey. *ACM Comput. Surv.* 56, 4 (2024), 102:1–102:37. doi:10.1145/3626521
[65] Xi Peng, Jiashi Feng, Jiwen Lu, Wei-Yun Yau, and Zhang Yi. 2017. Cascade Subspace Clustering. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA*, Satinder Singh and Shaul Markovitch (Eds.). AAAI Press, 2478–2484. doi:10.1609/AAAI.V31I1.10824
[66] Federico Piai, Paolo Atzeni, Paolo Merialdo, and Divesh Srivastava. 2023. Fine-grained semantic type discovery for heterogeneous sources using clustering. *VLDB J.* 32, 2 (2023), 305–324. doi:10.1007/S00778-022-00743-3
[67] Qi Qian. 2023. Stable Cluster Discrimination for Deep Clustering. *CoRR* abs/2311.14310 (2023). doi:10.48550/ARXIV.2311.14310 arXiv:2311.14310

[68] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21 (2020), 140:1–140:67. http://jmlr.org/papers/v21/20-074.html

[69] William M Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association* 66, 336 (1971), 846–850.

[70] Hafiz Tayyab Rauf, André Freitas, and Norman W. Paton. 2024. Deep Clustering for Data Cleaning and Integration. In *Proceedings 27th International Conference on Extending Database Technology, EDBT 2024, Paestum, Italy, March 25 - March 28*, Letizia Tanca, Qiong Luo, Giuseppe Polese, Loredana Caruccio, Xavier Oriol, and Donatella Firmani (Eds.). OpenProceedings.org, 636–649. doi:10.48786/EDBT.2024.55

[71] Kathy Razmadze, Yael Amsterdamer, Amit Somech, Susan B. Davidson, and Tova Milo. 2022. SubTab: Data Exploration with Informative Sub-Tables. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022*, Zachary G. Ives, Angela Bonifati, and Amr El Abbadi (Eds.). ACM, 2369–2372. doi:10.1145/3514221.3520154

[72] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics. doi:10.18653/v1/d19-1410

[73] Dominique Ritze and Christian Bizer. 2017. Matching Web Tables To DBpedia - A Feature Utility Study. In *Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017*, Volker Markl, Salvatore Orlando, Bernhard Mitschang, Periklis Andritsos, Kai-Uwe Sattler, and Sebastian Breß (Eds.). OpenProceedings.org, 210–221. doi:10.5441/002/edbt.2017.20

[74] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.* 115, 3 (2015), 211–252. doi:10.1007/S11263-015-0816-Y

[75] Alieh Saeedi, Eric Peukert, and Erhard Rahm. 2018. Using Link Features for Entity Clustering in Knowledge Graphs. In *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings (Lecture Notes in Computer Science, Vol. 10843)*, Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam (Eds.). Springer, 576–592. doi:10.1007/978-3-319-93417-4_37

[76] Gerard Salton, Anita Wong, and Chung-Shu Yang. 1975. A Vector Space Model for Automatic Indexing. *Commun. ACM* 18, 11 (1975), 613–620. doi:10.1145/361219.361220

[77] Erich Schubert. 2023. Stop using the elbow criterion for k-means and how to choose the number of clusters instead. *SIGKDD Explor.* 25, 1 (2023), 36–42. doi:10.1145/3606274.3606278

[78] Li Sun, Feiyang Wang, Junda Ye, Hao Peng, and Philip S. Yu. 2023. CONGREGATE: Contrastive Graph Clustering in Curvature Spaces. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*. ijcai.org, 2296–2305. doi:10.24963/IJCAI.2023/255

[79] Yuroti Tsuboi and Nobutaka Suzuki. 2019. An Algorithm for Extracting Shape Expression Schemas from Graphs. In *Proc. ACM Symposium on Document Engineering 2019*. 32:1–32:4. doi:10.1145/3342558.3345417

[80] Wenxuan Tu, Sihang Zhou, Xinwang Liu, Xifeng Guo, Zhiping Cai, En Zhu, and Jieren Cheng. 2021. Deep Fusion Clustering Network. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 9978–9987. doi:10.1609/AAAI.V35I11.17198

[81] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. journal of Machine Learning Research 9. *Nov (2008)* (2008).

[82] Ning Wang and Huaxi Liu. 2018. Annotating Web Tables with the Crowd. *Comput. Informatics* 37, 4 (2018), 969–991. doi:10.4149/CAI_2018_4_969

[83] Jianlong Wu, Keyu Long, Fei Wang, Chen Qian, Cheng Li, Zhouchen Lin, and Hongbin Zha. 2019. Deep Comprehensive Correlation Mining for Image Clustering. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 8149–8158. doi:10.1109/ICCV.2019.00824

[84] Renzhi Wu, Sanya Chaba, Saurabh Sawlani, Xu Chu, and Saravanan Thirumuruganathan. 2020. ZeroER: Entity Resolution using Zero Labeled Examples. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo (Eds.). ACM, 1149–1164. doi:10.1145/3318464.3389743

[85] Yihai Xi, Ning Wang, Shuang Hao, Yiyi Zhang, and Xinyu Chen. 2023. Popularity sensitive and domain-aware summarization for web tables. *Inf. Sci.* 621 (2023), 729–748. doi:10.1016/J.INS.2022.11.103

[86] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. 2016. Unsupervised Deep Embedding for Clustering Analysis. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016 (JMLR Workshop and Conference Proceedings, Vol. 48)*, Maria-Florina Balcan and Kilian Q. Weinberger (Eds.).

JMLR.org, 478–487. http://proceedings.mlr.press/v48/xieb16.html

[87] Bo Yang, Xiao Fu, Nicholas D. Sidiropoulos, and Mingyi Hong. 2017. Towards K-means-friendly Spaces: Simultaneous Deep Learning and Clustering. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017 (Proceedings of Machine Learning Research, Vol. 70)*, Doina Precup and Yee Whye Teh (Eds.). PMLR, 3861–3870. http://proceedings.mlr.press/v70/yang17b.html

[88] Linxiao Yang, Ngai-Man Cheung, Jiaying Li, and Jun Fang. 2019. Deep Clustering by Gaussian Mixture Variational Autoencoders With Graph Embedding. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 6439–6448. doi:10.1109/ICCV.2019.00654

[89] Xiaoyan Yang, Cecilia M. Procopiuc, and Divesh Srivastava. 2011. Summary Graphs for Relational Database Schemas. *Proc. VLDB Endow.* 4, 11 (2011), 899–910. http://www.vldb.org/pvldb/vol4/p899-yang.pdf

[90] Yaming Yang, Ziyu Guan, Zhe Wang, Wei Zhao, Cai Xu, Weigang Lu, and Jianbin Huang. 2022. Self-supervised Heterogeneous Graph Pre-training Based on Structural Clustering. In *NeurIPS*. http://papers.nips.cc/paper_files/paper/2022/hash/6c7297baffe5c85ea1d9e1ccb1222ab8-Abstract-Conference.html

[91] Yi Yang, Dong Xu, Feiping Nie, Shuicheng Yan, and Yueting Zhuang. 2010. Image Clustering Using Local Discriminant Models and Global Integration. *IEEE Trans. Image Process.* 19, 10 (2010), 2761–2773. doi:10.1109/TIP.2010.2049235

[92] Cong Yu and H. V. Jagadish. 2006. Schema Summarization. In *Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, September 12-15, 2006*, Umeshwar Dayal, Kyu-Young Whang, David B. Lomet, Gustavo Alonso, Guy M. Lohman, Martin L. Kersten, Sang Kyun Cha, and Young-Kuk Kim (Eds.). ACM, 319–330. http://dl.acm.org/citation.cfm?id=1164156

[93] Alexandros Zeakis, George Papadakis, Dimitrios Skoutas, and Manolis Koubarakis. 2023. Pre-trained Embeddings for Entity Resolution: An Experimental Analysis. *Proc. VLDB Endow.* 16, 9 (2023), 2225–2238. doi:10.14778/3598581.3598594

[94] Alexandros Zeakis, George Papadakis, Dimitrios Skoutas, and Manolis Koubarakis. 2025. An in-depth analysis of pre-trained embeddings for entity resolution. *VLDB J.* 34, 1 (2025), 5. doi:10.1007/S00778-024-00879-4

[95] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. 1996. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*, H. V. Jagadish and Inderpal Singh Mumick (Eds.). ACM Press, 103–114. doi:10.1145/233269.233324

[96] Yuchao Zhang, Yuan Yuan, and Qi Wang. 2023. Multi-level Graph Contrastive Prototypical Clustering. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*. ijcai.org, 4611–4619. doi:10.24963/IJCAI.2023/513

[97] Sheng Zhou, Hongjia Xu, Zhuonan Zheng, Jiawei Chen, Zhao Li, Jiajun Bu, Jia Wu, Xin Wang, Wenwu Zhu, and Martin Ester. 2022. A Comprehensive Survey on Deep Clustering: Taxonomy, Challenges, and Future Directions. *CoRR* abs/2206.07579 (2022). doi:10.48550/arXiv.2206.07579 arXiv:2206.07579

Received October 2024; revised January 2025; accepted February 2025