Deep Clustering for Data Cleaning and Integration

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.
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

ABSTRACT

Deep Learning (DL) techniques now constitute the state-of-the-art for important problems in areas such as text and image processing, and there have been impactful results that deploy DL in several data management tasks. Deep Clustering (DC) has recently emerged as a sub-discipline of DL, in which data representations are learned in tandem with clustering, with a view to automatically identifying the features of the data that lead to improved clustering results. While DC has been used to good effect in several domains, particularly in image processing, the potential of DC for data management tasks remains unexplored. In this paper, we address this gap by investigating the suitability of DC for data cleaning and integration tasks, specifically *schema inference*, *entity resolution* and *domain discovery*, from the perspective of tables, rows and columns, respectively. In this setting, we compare and contrast several DC and non-DC clustering algorithms using standard benchmarks. The results show, among other things, that the most effective DC algorithms consistently outperform non-DC clustering algorithms for data integration tasks. Experiments also show consistently strong performance compared with state-of-the-art bespoke algorithms for each of the data integration tasks.

1 INTRODUCTION

Deep Learning (DL) is now a well-established machine learning paradigm that is effective in domains as diverse as image processing [35], natural language processing [40], autonomous systems [39] and robotics [7]. DL is also the subject of extensive investigation for data management tasks, including those relating to data cleaning and integration [63].

Deep Clustering (DC) is a sub-domain of DL in which deep neural networks are used to learn data representations in tandem with a clustering algorithm in an unsupervised manner. DC jointly optimizes the representation learning and clustering [46]. The importance of deep clustering is increasing due the need to deliver representation paradigms that can operate over increasingly heterogeneous and high-dimensional datasets [2, 20]. DC also avoids the need for separate feature extraction, reduction and clustering [2].

DL has been applied successfully to a variety of data cleaning and integration problems, and several such problems involve clustering, so it seems timely to investigate the application of DC in data preparation. The approach in this paper is to empirically evaluate three DC algorithms, comparing them to baselines that use non-deep clustering techniques. For each of several problems, specifically *schema inference*, *entity resolution* and *domain discovery*, we: (i) define these tasks as clustering problems; (ii) identify several representations relevant to the tasks considering the type of data; (iii) compare the performance of three DC algorithms against three representative non-deep clustering algorithms; (iv) analyze the results in terms of overall quality and drill down to understand the behavioural properties of different techniques; and (v) compare the best performing DC algorithm with state-of-the-art bespoke solutions in each of the identified problems.

The contributions of this paper are as follows:

(1) The identification of DC as a promising approach for data cleaning and integration tasks that stand to benefit from clustering.
(2) The application of DC algorithms to *schema inference*, *entity resolution* and *domain discovery*, using vector representations for tables, rows and columns, respectively.
(3) An empirical evaluation using third-party benchmarks that shows that the most effective DC algorithms consistently outperform both non-DC clustering algorithms and state-of-the art bespoke algorithms in the areas from (2).

The remainder of this paper is structured as follows. Section 2 outlines the development of work on DC. Section 3 introduces key concepts in DC and describes the algorithms used in the experiments. Section 4 describes the experimental methods applied in the paper. Sections 5, 6 and 7 present experiments on different clustering methods for schema inference, entity resolution and domain discovery, respectively. Section 8 reflects on these experiments. Section 9 compares DC with bespoke algorithms. Section 10 presents some conclusions and areas for further work.

2 BACKGROUND AND RELATED WORK

This section briefly reviews related work on DC and discusses the components of DC, such as representation learning and clustering. Furthermore, we review how both modules can be optimized in a single framework when applied to data integration problems.

Standard clustering (SC) methods have achieved significant success for various applications when the data is low dimensional and where there is the assumption that vectors in the latent space are well-shaped and, most of the time, linearly separable. However, SC methods struggle to effectively perform clustering without representation learning when the data is unstructured, high-dimensional, and heterogeneous [77]. DC focuses on the joint optimization of high dimensional data representation in the latent space with suitability for clustering [77]. DC enables interaction between (i) clustering and (ii) representation learning through joint optimization to improve both of them iteratively.

Several proposals for clustering and representation learning architectures have been developed [77]. The representation learning architectures take a raw high dimensional embedding matrix as input and map it to a low dimensional latent space.

Series ISSN: 2367-2005

10.48786/edbt.2024.55

The most widely used representation learning architecture in deep clustering is Auto-encoder (AE) based unsupervised learning [31, 61]. The encoder function f_e encodes the input representation x_i into a low dimensional representation $h_i = f_e(x_i) = \frac{1}{1+e^{-(Wx_i+b_i)}}$, and the decoder function f_d decodes $f_e(x_i)$ into the reconstructed input $\overline{x}_i = f_d(h_i)$. W and b_i are the weights and bias of neural networks. The optimization function of a simple AE architecture for N samples can be defined as: $f_{min} = min\frac{1}{N}\sum_{i=1}^{N} \|x_i - \overline{x}_i\|^2$. Considering different applications, researchers proposed enhanced versions of AE, including Convolutional AE [21] for image clustering, Variational AE [68] for text classification, Generative AE [71] for image reconstruction, and Adversarial AE [53] to detect generative probabilistic novelty.

Feature distribution in the latent space is important; the learning efficiency depends on the features' distribution. In this context, subspace representation learning [13, 34, 73, 76] has been used widely for clustering. In subspace representation learning, the latent spaces are divided into several subspaces to categorize instances, and two instances are associated with linear relationships in the same subspace.

Regarding the clustering architecture in DC, it takes the optimized low-dimensional latent representation as input and returns the clustering soft assignments. At this stage, the learned representations are evaluated to determine whether it is more cluster-friendly, for example, if two contextual instances are close to each other in latent space. Several clustering techniques have been used in deep clustering [77]. The basic structure of the clustering component is to feed the d-dimensional representation using neural networks in the forward direction and reduce the dimensions to cluster number K. Then, a softmax layer can be used for the cluster assignment [77].

To bridge the semantic gap between representation learning and clustering, relation-matching deep clustering techniques have been used [19, 26], though such proposals are computationally expensive [77]. A further proposal uses graph-based architectures (e.g., [9, 43, 75]) with multiple distributions generated and fed to graph neural networks to preserve the hidden relations between the latent and K-dimensional target distribution.

3 DEEP CLUSTERING CONCEPTS AND TECHNIQUES

The fundamental difference between SC and DC is that SC methods act on a static representation, and DC methods adapt and learn the representation used for clustering. Most SC methods follow a hard clustering mechanism that takes a distance matrix as input and returns the 1-dimensional discrete clustering labels [77]. It is hard to optimize a 1-dimensional discrete vector for a neural network. Instead, DC methods work on a soft clustering mechanism that takes a high dimensional embedding matrix as input, learns the representation in a low dimensional latent space, and returns a K-dimensional continuous vector in the label space. The resulting K-dimensional continuous vector can be optimized for the final clustering 1-dimensional discrete vector [77].

The basic DC framework consists of three main components, i.e., (i) learning representation architecture, (ii) reconstruction loss, and (iii) clustering loss.

Concerning (i), we adopt DC methods that use AE based and subspace representation learning architectures. A DC framework with a basic AE architecture is presented in Figure 1. In an AE based architecture, the clustering is based on the lower dimensional representation produced by the encoder, with a loss function that trades off cluster quality with the ability to reconstruct the original representation. Through this approach, the latent space representation of the original input data should preserve the most suitable features for clustering.

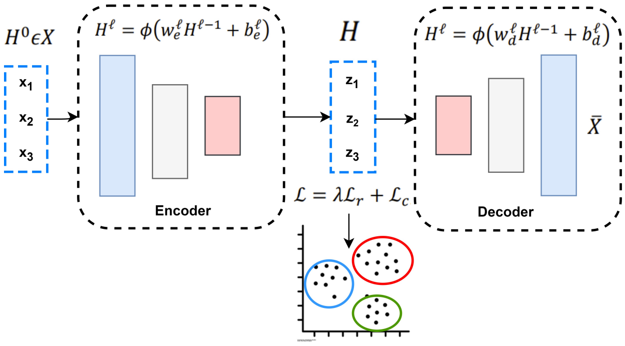

$H^0 \epsilon X$
x_1
x_2
x_3
$H^\ell = \phi(w_e^\ell H^{\ell-1} + b_e^\ell)$
Encoder
H
z_1
z_2
z_3
$\mathcal{L} = \lambda\mathcal{L}_r + \mathcal{L}_c$
$H^\ell = \phi(w_d^\ell H^{\ell-1} + b_d^\ell)$
Decoder
$\bar{X}$

Figure 1: DC with basic AE architecture

Consider raw data $X\epsilon\mathcal{R}^{N\times d}$, where $\mathcal{R}^{N\times d}$ belongs to a d- dimensional embedding matrix $\mathcal{R}$ with N elements, x_i is the ith element in X. Representation learning in AE initiates with the encoder part, the purpose of which is to encode X into a low dimensional latent representation H. Let's suppose the AE consists of L layers where ℓ is layer number, the initial representation learned from $\mathcal{R}^{N\times d}$ in encoder H^ℓ can be obtained as [9]:

$$H^\ell = \phi\left(w_e^\ell H^{\ell-1} + b_e^\ell\right), \tag{1}$$

where $H^0\epsilon X$ and ϕ denotes the activation function, w_e^ℓ and b_e^ℓ represents the weight and bias of ℓth layer. The decoder part decodes H into reconstructed input $\overline{X}$ using the following equation [9]:

$$H^\ell = \phi\left(w_d^\ell H^{\ell-1} + b_d^\ell\right), \tag{2}$$

where $H^0\epsilon\overline{X}$, w_d^ℓ and b_d^ℓ represents the weight and bias of ℓth layer for decoder. The objective function used in AE architectures can be defined as:

$$\mathcal{L} = \lambda\mathcal{L}_r + \mathcal{L}_c, \tag{3}$$

where $\mathcal{L}_r$ and $\mathcal{L}_c$ represent reconstruction and clustering loss, respectively. $\mathcal{L}_c$ is clustering module specific and each deep clustering proposal provides several other module specific losses that are combined with $\mathcal{L}_c$. The basic version of $\mathcal{L}_r$ is:

$$f_{min} = min\frac{1}{N}\sum_{i=1}^{N}\left\|X - \overline{X}\right\|^2 \tag{4}$$

We adopted three recently proposed DC algorithms, SDCN [9], EDESC [13] and SHGP [69], to evaluate on data integration tasks. The selection of the DC algorithms is based on their implementation suitability and the flexibility of their proposed distance functions towards data integration tasks; many DC methods are purely designed for image and text clustering applications [50, 64, 67] and are not obviously suitable for comparing rows, columns, and tables in the latent space. Another selection criterion is performance; we describe the top three performers on data integration tasks. The description of the DC algorithms used for the experimental evaluation is given below.

SDCN [9] is based on two representation learning modules, a Graph Convolutional Network (GCN) and an AE, that work in parallel to learn structural and AE specific information. SDCN starts by constructing a K-Nearest Neighbor (KNN) graph from X and feeding it to the GCN model to learn the structural information. To learn AE-specific representations, SDCN uses a simple AE architecture. GCN-specific and AE-specific representations are combined through a delivery operator and dual self-supervised mechanism to perform soft clustering assignments from multiple representations.

EDESC [13] is a deep subspace clustering method. Subspace representation learning involves mapping data points into low dimensional subspaces to separate each data point, similar to the early stage of subspace clustering [77]. Unlike SDCN, EDESC is not graph-based. Deep subspace clustering models are self-expressive and assume a linear combination between one data point and its other data points from the same subspace. The simplest self-expressiveness property can be denoted as $X = XC$, where X represents the data matrix, and C represents the self-expression coefficient matrix. The objective function for self-expression-based representation learning can be defined as [77]:

$$\min_{C} \|C\|_p + \frac{\lambda}{2} \|H - HC\|_F^2 \quad s.t. \quad diag(C) = 0 \tag{5}$$

where $\|.\|_p$ shows matrix norm and λ is weight controlling factor. H is the representation learned by the network. EDESC takes a deep representation and learns the subspace bases in an iterative refining manner. The latent space representation is learned through the refined subspace bases outside the self-expressive framework. EDESC initializes a subspace D using K-means [13].

SHGP [69] employs self-supervised learning on Heterogeneous Information Networks (HINs) and uses a combination of attention-aggregation schemes using two modules, *Att-LPA* and *Att-HGNN*, that improve each other to construct and learn object embeddings. The *Att-LPA* module generates pseudo-labels, serving as a self-supervised signal to guide the learning process. *Att-LPA* uses a structural clustering method (LPA), which assigns and iteratively refines the labels to objects. These pseudo-labels are then utilized as a guide to enhance the learning of object embeddings and attention coefficients in the *Att-HGNN* module. *Att-HGNN*, directed by the pseudo-labels, uses object features and attention coefficients to combine information from neighboring features and learns the embeddings effectively. For the clustering task, SHGP uses K-means on the embeddings produced by both modules.

SDCN and EDESC build on a pre-trained AE that learns a compressed representation of the optimized input embedding for reconstruction while ignoring the clustering task. AE can help reduce the input embedding's dimensionality, remove noise and redundancy, and capture relevant patterns and structure. Then, the learned representation passes to the original training part combined with clustering loss for further fine-tuning. It is helpful to evaluate the impact of learned representation on non-DC algorithms. In this context, we used a different AE version that employs the Birch and K-means algorithms to perform clustering not directly on the embedding but on the representation learned by AE. This can be interpreted as performing Birch and K-means on H in Figure 1.

4 EXPERIMENTAL SETUP

The hypothesis is that DC is expected to outperform SC as it builds on a latent space representation that can better integrate schema-level and instance-level representations. To evaluate the hypothesis, we included the following SC methods.

K-means [29] initializes with a set of data points; in the context of the data integration problem, it initializes with distance vectors of either schema or instance-level data points in the vector space and assigns the data points to the clusters with the nearest centroid. It repeatedly iterates to optimize the cluster centers. K-means minimizes the clustering loss with a squared Euclidean distance function. K-means requires the value of K in advance to predict the clusters.

Birch [74] is a hierarchical clustering algorithm designed for tackling large databases, especially involving noise and outliers. Birch is supervised in terms of number of clusters K. Birch provides a hierarchical clustering structure, which can help understand the data structure and provide more interpretable results.

DBSCAN [24] is a density-based spatial clustering algorithm primarily designed for large databases with noise. DBSCAN identifies clusters based on the density of the data; it has two main parameters: the radius ε, which defines the area in the neighboring points, and *MinPts*, which represents a minimum number of points required to declare the area dense enough to form a cluster. Unlike K-means and Birch, DBSCAN is suitable for identifying clusters with irregular shapes and does not require specifying the number of clusters K in advance. DBSCAN is sensitive to its parameters ε and *MinPts* and tends to provide poor clustering without optimizing its parameters. In our experiments, we used a commonly used heuristic method called the elbow method [58] to identify ε. In the elbow method, the distances of data points to their nearest neighbors are calculated and plotted on a graph, and the "elbow" point is the best ε-value where the curve intersects. The *MinPts* are set to the total number of clusters K when the rule of thumb (*MinPts* = 2×dim, where dim= the data dimensions) does not work.

Setting the number of clusters K in advance gives the SC methods an (in a sense unfair) advantage as they are given the (unknown) Ground Truth (GT) value for K. In contrast, DC methods only take K to initialize the centers of the clusters for pre-training. Subsequently, the DC methods work out K without taking a GT value in training, and in practice, it may not be possible to establish the correct K. As such, the DC methods are more flexible in their ability to automatically estimate the number of clusters without the need for prior knowledge of K.

We used the scikit-learn implementations [51] of the SC algorithms. A detailed overview of the experimental framework is presented in Figure 2, which consists of three main phases from left to right. Firstly, the raw data is preprocessed to remove high-level syntactic errors. In the second phase, the preprocessed data is fed to the embedding module to generate dense representations. Lastly, dense representations are further enhanced in the clustering module and final clustering assignments are produced.

4.1 Evaluation Metrics

We employ two widely used standard clustering evaluation metrics, Accuracy (ACC) [70] and Adjusted Rand Score (ARI) [66].

ARI can be defined as [66]: Assume we are given a set S of n elements and two clustering sets of these elements consists of r and s groups represented as $X = \{X_1, X_2, \ldots, X_r\}$ and $\{Y = Y_1, Y_2, \ldots, Y_s\}$ in a contingency Table $[t_{ij}]$ of overlaps between X and Y. Each element in $[t_{ij}]$ shows the count of common objects between X_i and Y_j. ARI can be defined from $[t_{ij}]$:

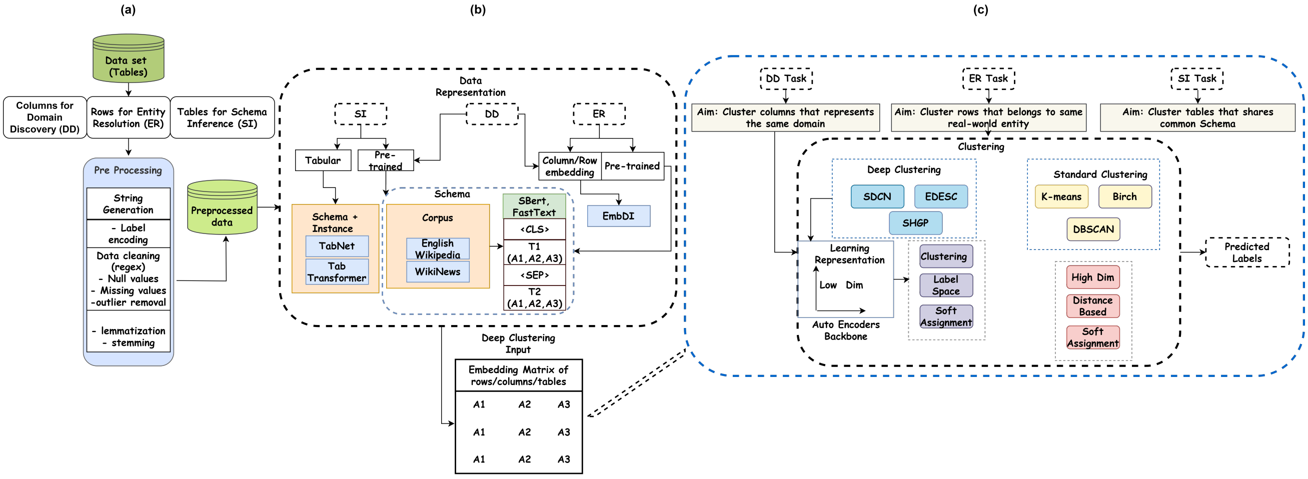


Figure 2: Overview of the experimental framework. (a) represents raw data cleaning phase, for example, handling missing or infinite values in tables, (b) is for getting the embedding matrix from tables, rows, or columns, (c) represents the DC module, where the input will be an embedding matrix from (b) for each data integration problem.

$\left[t_{ij}\right]$ can be represented as:

$$\left[t_{ij}\right] = \begin{array}{c|cccc|c}
{}_{X}\backslash^{Y} & Y_1 & Y_2 & \dots & Y_s & Sums \\
\hline
X_1 & t_{11} & t_{12} & \dots & t_{1s} & a_1 \\
X_2 & t_{21} & t_{22} & \dots & t_{2s} & a_2 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
X_r & t_{r1} & t_{r2} & \dots & t_{rs} & a_r \\
\hline
Sums & b_1 & b_2 & \dots & b_s &
\end{array}$$

$$ARI = \frac{\sum_{ij}\binom{t_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_i}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_i}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_i}{2}\right] / \binom{n}{2}} \tag{6}$$

ARI determines the similarity between two clustering results; usually, one is GT labels, and the second corresponds to the labels returned by the clustering algorithm. Generally, the value of ARI lies between 0 and 1. An ARI value closer to 1 represents a strong match between predicted and GT clusters.

The clustering ACC for N samples, with cluster id R $\in c_i$ and GT id $T \in gt_i$ can be defined as:

$$ACC\,(R, T) = \frac{\sum_{i=1}^{N} \delta(gt_i,\ map(c_i))}{N} \tag{7}$$

$$\delta\left(gt_i,\ map\left(c_i\right)\right) = \begin{cases} 1, & if\ gt_i = map\left(c_i\right) \\ 0, & otherwise \end{cases} \tag{8}$$

Function $map()$ gives the best permutation mapping between predicted and GT labels through the Hungarian Algorithm [12]. ACC maps the predicted labels into ground labels since cluster ids in the prediction are randomly generated and dissimilar to those assigned to GT labels.

4.2 Hyper-parameter setting

Network learning benefits from hyper-parameters optimized for the particular task. Since deep clustering algorithms are heavily used for image processing tasks, optimizing hyper-parameters for data integration tasks is necessary. In this context, we have four basic parameters of SDCN, EDESC and SHGP:

The number of layers are important because they determine the network's ability to learn unsupervised feature representations. Since we have pre-defined embeddings as AE inputs, rows, columns or tables with similar meanings are positioned closer together. An AE with fewer layers can efficiently compress and reconstruct this underlying structure. We fixed *number of layers = 2* in all experiments of SDCN and EDESC after experimenting with different values. However, SHGP uses *Att-LPA* and *Att-HGNN* with several layers from graph neural networks, so we used the default number of layers based on the SHGP paper, i.e., two *Att-HGNN* encoder layers and several hidden layers in the set 64, 128, 256, 512.

Layer size (refers to the number of neurons in each layer) provides the capacity and ability of AE to learn complex patterns in the data. In order to maintain a high-dimensional hidden representation, we fixed *layer size = 1000* for all experiments of DC methods after experimenting with different values. This suggests that the complexity of row, column, or table embeddings requires a larger hidden layer size to retain more semantic information.

Latent space size z: Originally, SDCN and EDESC used $z = 10$, but it is too small in data integration tasks to capture the complexity of the row, column or table embeddings, leading to significant information loss. Considering this, after systematically experimenting with different values, we fixed z*=100* for SDCN and AE, and $z = a$ for EDESC where the shape of $a = (n\_clusters \times d)$, and d represents the dimension of the subspace. For SHGP, the size of the learned representation depended on the size of label space and appeared optimal after experimental evaluation as $z = s$ where s *= size of label space.*

Training Epochs: We used the silhouette coefficient [56] on the learned representation with predicted clusters to choose where to stop training. We pre-train SDCN and EDESC for 30 epochs except for entity resolution (100 epochs), which requires more pre-training due to the large numbers of clusters. We decide the number of training epochs based on the best silhouette score. Since SHGP uses K-means for clustering the learned embedding, the embeddings obtained on 50 and above epochs do not significantly impact K-means clustering. We used fixed 50 epochs for representation learning by *Att-LPA* and *Att-HGNN.*

It is evident from the related work [69] that AE (or similar DC architectures) applied with an SC algorithm produces good results. Further examples of combining SC methods with AE include H-DC [69] and DeepCluster [15]. We use AE (described in Section 3) with Birch for the entity resolution and domain discovery experiment instead of SDCN. The learned features without clustering loss from the AE step were more effective at capturing the underlying structure of the data than fine-tuning features along with clustering loss. Regarding the decision as to which clustering to choose in this setting, we used the silhouette score as an unsupervised way of evaluating cluster quality. If the silhouette score converges during training with SDCN, we use SDCN; otherwise, we retain AE and perform clustering using Birch. The hyperparameter settings for the experiments, along with the source code, are available[1].

5 SCHEMA INFERENCE

Schema inference proposes a schema that makes recurring structural features in data explicit. Schema inference may be applied to extensional data (e.g., inferring a JSON schema from several JSON documents [5]) or to intensional data (e.g., inferring a schema that summarises a complex relational database [72]). Schema inference has been a topic of ongoing investigation for different data models, and several surveys have been produced [16, 36, 38]. It is common for schema inference to build on clustering, to identify candidate types/classes in the data, so clustering is an important enabling technology for schema inference. We consider schema inference as a clustering problem, where the task is: for a given set of datasets $D = \{d_1, d_2, \ d_3 \ldots d_n\}$ identify every subset $D_s \subseteq D$ that can share a common schema using clustering.

Pre-trained embeddings have been used widely for data integration tasks [22]. They fall into two specific categories: sentence-based and word based. Sentence-based embeddings directly map a sequence of tokens into a single dense vector. In contrast, word-based embeddings encode each token separately, and then perform an aggregation function to derive a single vector. Pre-trained embeddings are trained on large corpora and tend to have broader vocabulary coverage. Considering this, we choose two pre-trained embeddings (one word-based, FastText [28], and one sentence-based, SBERT [54]) to perform schema inference with schema-level evidence. When carrying out schema inference, the schema-level information includes only table headers; each table is represented by a string (combination of attribute names). Nevertheless, pre-trained embeddings have disadvantages when tackling large databases, especially with instance level data, e.g., where there is specialised vocabulary, or numerical data distributions [14]. To produce embeddings with Schema+instance-level data, tabular transformers have been a mainstream option in the deep learning community [6]. Several tabular transformers are proposed in the literature to handle noisy and incomplete data (e.g., [4, 27, 32, 60]).

For evaluation, we use the T2D Entity-Level Gold standard (T2D) [55] web table dataset and the Table Union Search (TUS) benchmark [47] that identifies tables that are unionable. In T2D, we rejected all tables that included languages other than English. We also excluded all DBpedia *Thing* tables to avoid significant data imbalance, as it is mapped to more than 50% of data tables. In the TUS benchmark, we aim to determine which tables from a set can be unioned together, and we set the criteria that two tables are unionable if at least 40% of their corresponding attributes are unionable. We cluster the tables using the Louvain community detection algorithm [8]. Each detected community, representing a group of unionable tables, and is assigned a unique GT label. We excluded all single-table communities. Further properties of the web tables data and TUS benchmark are given in Table 1. The criteria for choosing a tabular transformer are based on the nature of the datasets. Web tables are noisy; for example, a table with attribute: symbol and values 'aa', 'axp' is problematic for pre-trained embeddings as 'aa', 'axp' are not present in the pre-trained vocabulary, and most of the cases will be treated as unknown tokens. However, training on a local vocabulary can overcome this issue. Another significant issue is incomplete columns or rows. To handle these issues, we evaluated several transformers, including Tabnet [4], TabTransformer [32], SAINT [60], FT-Transformer [27], TabFastFormer [37], TabPerceiver [33] and EmbDi [14], on web tables and TUS. We retained the two best performers, Tabnet and TabTransformer for comparison. TabTransformer [32] has been found to be robust with missing table values and noisy data. TabTransformer is based on Transformers [65] with several multi-head attention layers to contextually embed categorical columns. Tabnet [4] is based on row-wise feature selection and is more suitable for raw data without pre-processing. Tabnet uses sequential attention to choose categorical and numerical features at each decision step.

5.1 Data dimensionality for tabular transformers

To produce an embedding matrix (X_i), tabular transformers give each table a different dimension size d. In our experiments, we use the standard values of d for FastText and SBERT as 300 and 768, respectively. Tabnet and TabTransformer process each input feature individually, whether row or column and apply a series of transformations. Each table's categorical and continuous features have different cardinalities affecting the size of output embedding d for each table in Tabnet and TabTransformer. To normalize d for instance-level data, we selected the maximum feature size occurrence and performed linear interpolation to fill the empty values. However, for TabTransformer, the last column of the embedding matrix needs the preceding value to interpolate, which makes the size of d as max(d−1), where $max(d)$ denotes the maximum number of dimensions observed for any table. The obtained values of d for Tabnet and TabTransformer are 693 and 208 for web tables 1365 and 352 for TUS data, respectively.

5.2 Results and Discussion

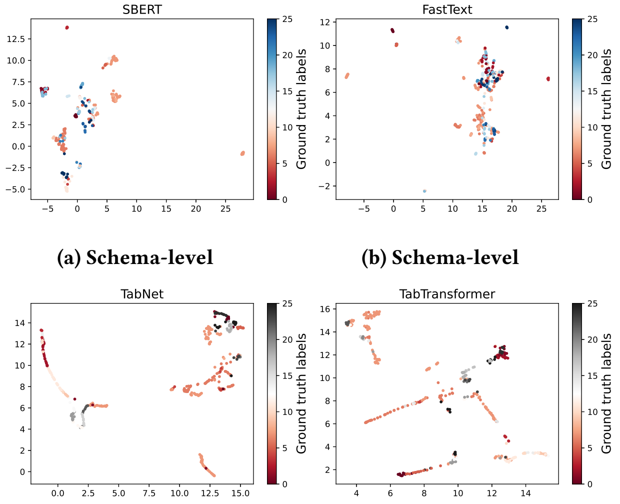
For all experimental results, the bold and underlined values in the tables indicate the best and the second-best results considering the corresponding embedding methods, respectively. Table 2 presents the clustering results for schema inference using only schema-level data. The following can be observed:

(i) **The selected table representation significantly impacts performance, with SBERT significantly outperforming FastText in most cases for all clustering algorithms on all datasets**. SDCN achieved 0.38 higher ARI with SBERT than FastText, where Birch and K-means with SBERT outperformed Fast Text by 0.34 and 0.17 in the ARI on web tables data. Similarly, EDESC and SHGP with SBERT are superior by 0.23 and 0.13 ARI to FastText on TUS data. Figures 3a and 3b confirm that the separability of data points for SBERT is more robust than for FastText, in which data points are compact in the latent space,

[1] https://github.com/hafizrauf/dc_data-integration

Table 1: Dataset properties for schema inference, entity resolution and domain discovery

Properties	Schema Inference		Entity Resolution		Domain Discovery	
	web tables	TUS	Music Brainz 2K	GeoSet	Camera	Monitor
Sources	N/A	N/A	5	4	24	26
Number of Instances	429	4248	$\sim 2K$	3021	19036	34481
GT clusters	26	37	684	786	56	81



(a) Schema-level

(b) Schema-level

(c) Schema+Instance-level

(d) Schema+Instance-level

Figure 3: Umap representation [45] of pre-trained sentence and tabular based encodings on web tables data. X and Y axes represent the dimensions of the reduced UMAP space. UMAP preserves the local structure, showing that points that were close together in the embedding space are near each other in the UMAP space.

which is unsuitable for clustering. (ii) **The DC algorithms outperform the SC algorithms in most cases, with the largest differences being for SBERT-supported models**. EDESC with SBERT obtained higher ARI scores (0.15, 0.18, and 0.66) on TUS data compared to K-means, DBSCAN and Birch, respectively. K-means, which considers convex and isotropic clusters, obtained a modest ACC difference of 0.09, signifying its likely struggles with dense data. Birch did better, and achieved an ARI of 0.33, but still lower than SDCN (0.13 ARI) and EDESC (0.08 ARI) on web tables data. DBSCAN predicted one cluster with zero ARI, which shows its lower ability with varying data densities. (iii) **SDCN predicted fewer clusters than other competitors while maintaining superior performance.** SDCN with SBERT predicted 16 clusters, diverging from the 26 predicted by EDESC and SHGP, matching the GT count. However, SDCN's superior performance compared to EDESC and SHGP indicates its AE's ability to prioritize clusters' internal coherence and quality over count. In contrast, EDESC and SHGP failed to maintain the inter-cluster separability, even when meeting the GT count.

Table 3 presents the clustering results for schema inference using both schema and instance level data. The following can be observed: (i) **SDCN is significantly more compatible with Tabnet than TabTransformer on all datasets**. SDCN with Tabnet obtained an ARI score 0.19 and 0.13 higher than TabTransformer on web tables and TUS data, respectively. This superior performance exhibits Tabnet's ability to select prominent features at each decision step, thus improving SDCN's feature extraction process with more detailed data understanding. Figures 3c and 3d represent no significant latent space difference in terms of data points' relative positions. This low visual difference entirely depends on the Umap projections, which can lead to the loss of information in relationships among the tables. Furthermore, it shows that web table data does not have a clear cluster structure, making it difficult to discover meaningful patterns. Adding instance-level evidence with tabular embedding failed to show its suitability for clustering compared with schema-level evidence with SBERT for both datasets. (ii) **Changing the embeddings does not affect the overall performance trend for the DC method when we consider Schema+Instance-level data**. We observe that DC methods outperformed SC methods with both Tabnet (SDCN obtained 0.46, 0.45, 0.46 higher ARI compared to K-means, DBSCAN, and Birch, respectively) and TabTransformer (SDCN obtained 0.24 higher ARI compared to K-means, DBSCAN, and Birch) on web tables data. Like schema-level, in Schema+Instance-level, DBSCAN's repeated inability to effectively differentiate clusters within the dense data representation resulted in a single cluster for Tabnet and TabTransformer on web tables data. (iii) **The provision of K does not significantly impact the clustering algorithms' overall performance.** For example, Birch with TabTransformer may have been expected to outperform EDESC due to the prescription of a fixed number of clusters (26), but EDESC outperformed Birch by 0.07 ARI even though it only produced 14 clusters. Similar behavior can be seen when using Tabnet with EDESC, which produced 12 clusters compared to 26 GT clusters and achieved 0.09 higher ARI than K-means, which generated 26 clusters on web tables data. On the other hand, DBCSCAN produced 3 clusters on TUS data but failed to produce convincing clusters, compared EDESC with an exact number of predicted clusters 37 as GT. (iv) **Tabnet and TabTransformer treat all attributes as being equally important.** As a result, even when two tables share a subject attribute, they may be clustered separately because their other attributes are different. A *subject attribute* identifies the artifact that the table is about. For example in web tables data, tables T1 and T2, are clustered separately where they have a common subject column *Country* and other columns *(T1.Total population in 2004 (million), T1.Annual population growth rate (%), T1.Population density (persons per square km.), T1.Average number of persons per household)* and *(T2.rank, T2.population, T2.date of information)*.

In terms of relative performance between Tables 2 and 3, empirical results for the web tables dataset show that: **schema-level evidence is more suitable for DC and SC, and adding instance-level evidence leads to poorer performance**. This is because the actual instances tend to have low overlap even when their tables are clustered together in the GT. For example, SDCN with Tabnet failed to cluster tables T3 and T4, which belong to the class *Film* because of the same schema and different instances, e.g., (*T3.fansrank: 101, T3.title: treasure sierra madre,*

Table 2: Schema Inference: Schema-level clustering results DC (SDCN, EDESC and SHGP) vs SC (K-means, Birch and DBSCAN) using pre-trained embeddings on web tables and TUS datasets.

		SDCN		SHGP		EDESC		K-means		DBSCAN		Birch	
Dataset	**Metric**	SBERT	FastText	SBERT	FastText	SBERT	FastText	SBERT	FastText	SBERT	FastText	SBERT	FastText
	K	16	19	26	26	26	26	26	26	1	1	26	26
web	ARI	**0.46**	0.08	0.10	0.05	0.41	**0.14**	0.27	0.10	0.0	-0.018	0.33	-0.01
tables	ACC	**0.58**	0.27	0.32	0.27	0.55	**0.35**	0.45	0.31	0.29	0.24	0.49	0.28
	K	37	33	37	37	37	36	37	37	18	4	12	1
TUS	ARI	0.74	**0.70**	0.65	0.53	**0.88**	0.65	0.73	0.63	0.70	0.05	0.22	0.0
	ACC	0.79	**0.74**	0.73	0.63	**0.87**	0.73	0.79	0.69	0.78	0.29	0.40	0.20

Table 3: Schema Inference: Schema+Instance-level clustering results DC (SDCN, EDESC and SHGP) vs SC (K-means, Birch and DBSCAN) using tabular embeddings on web tables and TUS datasets. TT and TN refer to TabTransformer and Tabnet, respectively.

		SDCN		SHGP		EDESC		K-means		DBSCAN		Birch	
Dataset	**Metric**	TT	TN	TT	TN	TT	TN	TT	TN	TT	TN	TT	TN
	K	26	26	26	26	14	12	26	26	1	1	26	26
web tables	ARI	**0.26**	**0.45**	0.02	-0.019	0.09	0.08	0.02	-0.013	0.023	-0.007	0.02	-0.013
	ACC	**0.42**	**0.55**	0.29	0.25	0.31	0.31	0.29	0.27	0.29	0.26	0.28	0.27
	K	37	37	37	37	37	37	37	37	3	3	37	37
TUS	ARI	**0.29**	**0.34**	0.06	0.06	0.24	0.25	0.21	0.25	0.02	0.03	0.18	0.26
	ACC	**0.44**	**0.45**	0.21	0.21	0.40	0.38	0.38	0.38	0.26	0.26	0.35	0.38

T3.year: 1948, T3.director: john huston, T3.overallrank: 92) and *(T4.fansrank: 442, T4.title: game, T4.year: 1997, T4.director: david fincher, T4.overallrank: 1491).*

Overall, considering the observations and evidence from Tables 2 and 3, we can conclude that DC outperforms SC, regardless of the selected embedding strategy.

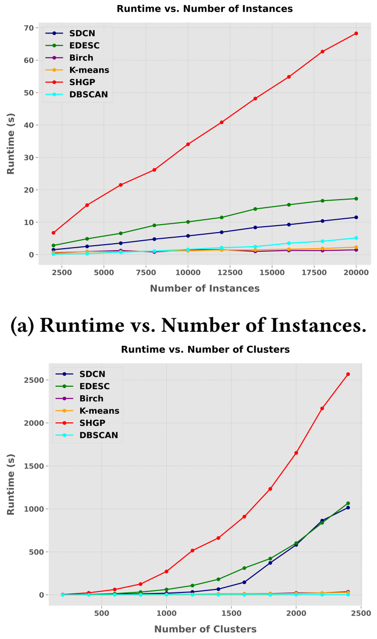
6 ENTITY RESOLUTION

Entity resolution is the well-studied process of identifying where two or more records in a dataset represent the same real world object [17, 23]. Entity resolution thus takes place at the instance level. Most entity resolution proposals focus on pairwise similarity between records. However, the transitive closure of the pairwise similarity relationship may not lead to suitable clusters, and as a result some entity resolution proposals include a clustering step (e.g., [18, 30]). We note that deep learning has been applied with positive results to entity resolution (e.g., [22, 62]), but that the need for training data is a barrier to adoption [11]. Deep clustering can potentially provide some of the benefits of deep representation learning in an unsupervised setting. The task of entity resolution with clustering can be defined as: given a set of records $R = \{r_1, r_2,\ r_3 \ldots r_n\}$ identify every subset $R_s \subseteq R$ that refers to the same real world entity using clustering.

For the entity resolution task, we employed the MusicBrainz [57] and Geographic Settlements (GeoSet) datasets [57]. MusicBrainz contains continuously updated song data from five sources and includes duplicates for 50% of the original records, whereas GeoSet contains geographical real-world entities from four data sources.

The original "Music Brainz 200K" version contains 100,000 GT clusters. As this dataset has a large number of clusters, we use it to investigate the scalability of the DC and SC algorithms, reporting algorithm runtime for varying numbers of clusters and instances (see Figure 4). To investigate the impact of the number of instances on performance, we hold $K = 200$ as a constant and duplicate the clusters to keep a fixed K for varying numbers of instances. Figures 4a shows that the run time of SC methods is significantly lower than that of DC methods, and that run time grows broadly linearly for all methods. Not surprisingly, the DC methods are slower, as DC involves deep neural networks with many hidden layers and parameters, and training of these networks contains the computation of gradients and the updating of network weights, which is computationally expensive. SHGP is several times slower than other DC methods because it uses structural clustering to generate pseudo-labels, and clustering from complex relationships and structures within heterogeneous graphs is time-consuming [69].

To investigate the impact of the number of clusters on performance (see Figure 4b), we choose the number of instances corresponding to different values of K. We can observe that increasing the number of clusters significantly impacts the run time of all DC methods. All DC methods have relatively low run times until about 1000 to 1500 clusters, at which point run times rapidly increase. In SDCN, the computational cost of distance calculations (from each data point n to its cluster centroid) grows linearly when the number of clusters is small, and the runtime is dominated by the number of instances. However, as the number of clusters increases, SDCN needs to do more computation from each data point to each cluster centroid, causing the runtime to grow more than linearly. SHGP uses K-means to get hard cluster labels from low-dimensional embeddings. When we increase K, the assignment step (where each data point is assigned to the nearest centroid) takes longer as there are more centroids to compare. Similarly, the process takes longer when the centroid is updated based on its assigned data points because there are more centroids to update. EDESC faces the same issues during the initialization of the subspace with K-means clustering. SC methods have linear runtime growth. K-means updates its centroid (distance calculation to each data point from each centroid) once per iteration and is not impacted by increasing K. DBSCAN does not depend on the K, but instead, the density of data, leading to the



(a) Runtime vs. Number of Instances.

(b) Runtime vs. Number of Clusters K.

Figure 4: Runtimes for different numbers of instances and clusters

linear runtime. Birch used data points to construct the clustering feature tree, and increasing K does not affect this construction, leading to linear runtime.

Considering the scalability issue of DC methods, we reduce the Music Brainz 20K [57] to Music Brainz 2K to provide more manageable run times for entity resolution tasks. To ensure the dataset is balanced, we discarded all instances associated with a single cluster, sorting them by cluster-ID in increasing order, and chose the top $\sim 2K$ instances with 684 clusters. The properties of the datasets evaluated for entity resolution are given in Table 1.

We use schema+instance-level data to cluster all records that describe the same real world entity. Schema-level information is not considered because each record in the Music Brainz 2K dataset contains the same attributes with different descriptions.

Embedding rows with data heterogeneity problems can be challenging, for example, coping with missing attributes for a particular record, the size of the description, and data type ambiguity (for example handling numeric data and multi-word tokens). Consider a scenario of identifying duplicate records with different descriptive patterns *(year:2008, language:eng)*, *(year:'08', language:English)*, *(year:, language:eng)* and *(year:2008, length: 24sec)*. The data suffer from several issues, including missing year, year value with numerical and categorical type, same record with different attribute and value abbreviations. Considering these issues, we used EmbDi [14] to embed records into the embedding matrix, which can be directly input to the DC algorithms. EmbDi [14] is based on a tripartite graph with three types of node, specifically value node (representation of unique value), a column node (corresponds to the columns or attribute representation), and a row node (a unique token for each tuple). These nodes are connected in a graph based on the structural information that exists in the dataset. EmbDi adopts random walks between neighboring nodes to capture the local and global structure in the graph, where the length of the random walk and the number of walks per node are user-defined. Column nodes with similar neighborhoods are placed together in the embedding space. EmbDi offers optimizations to handle data heterogeneity problems. We selected only those encodings produced by EmbDi with prefixes (see [14]) *idx_*, as each token with prefix *idx_* represents one tuple.

As SBERT has shown competitive performance on schema inference, we have also applied the pre-trained SBERT model to entity resolution tasks. We computed the SBERT embeddings of each row of the six attributes in the Music Brainz 2K and three attributes in GeoSet.

6.1 Results and Discussion

Table 4 presents the clustering results for entity resolution using Schema+Instance-level data. The following can be observed: (i) **Running SDCN did not manage to improve the representation compared to AE for any of the datasets.** We observed that SDCN was not further optimizing the representation learned by AE during pre-training as measured by the silhouette score. Due to this, we used the representation of both datasets learned by AE from the pre-training module without considering the clustering loss from SDCN. (ii) **Most clustering algorithms produced better results with SBERT than with EmbDi.** AE with SBERT leads with a 0.26 (for Music Brainz) and 0.24 (for GeoSet) higher ARI than AE with EmbDi. For Music Brainz data, AE with SBERT obtained 616 more *TP* pairs than AE with EmbDi. For example, one pair, which is *TP* in AE with SBERT and *FN* in AE with EmbDi, is *(title: 009-Ballade a donner, length: 4m 2sec, artist: Luce Dufault, album: Luce Dufault (1996), year: nan, language: Fre.)* and *(title: Luce Dufault - Ballade Ã donner, length: 242, artist: nan, album: Luce Dufault, year: 96, language: French)*. EmbDi encoded *(length: 242)* as numerical which is given in seconds and *(length: 4m 2sec)* as a string token, whereas SBERT considered both as strings. Similarly, EmbDi did not manage to preserve the contextual information by comparing text with its abbreviations *(language: Fre. vs. language: French)*. (iii) **The best overall results are with AE for both representations in DC methods.** AE outperforms EDESC on Music Brainz data with 0.08 and 0.34 higher ARI scores with EmbDi and SBERT, respectively, since AE learned features more effectively than those learned during the training with EDESC. ACC shows that AE with SBERT assigns 7% more samples to the correct clusters in the prediction compared to the number of clusters assigned with EDESC using SBERT. For example, the cosine similarity of two contextually similar SBERT vectors representing *(title: Uriah Heep - Southern Star, length: 266, artist: nan, album: Into the Wild, year: 11, language: English)* and *(title: 0B1-Southern Star, length: 4m 26sec, artist: Heep Uriah, album: Into the Wild (2011), year: nan, language: Eng.)* is 0.78, and should be clustered together with high contextual similarity. However, EDESC placed the two rows in separate clusters compared to AE, which produced the correct clusters. (iv) **EDESC with SBERT failed to distinguish most of the unary clusters (*TN* in GT) leading it to predict the incorrect number of clusters (668 against 684 GT clusters) compared to AE**. Most EDESC unary clusters have been merged in the prediction, causing a high *FP* rate (EDESC misassigns rows to the same cluster when they should be in different clusters). For example, two instances sharing lexically similar values *(length: 4m 56sec, year: nan, language: Eng.)* and *(length: 4m 29sec, year: nan, language: Eg.)* belonging to different clusters in GT but obtain high cosine similarity (0.99) in the EDESC latent space with SBERT resulting in a misclassification. (v) **The original EmbDi**

Table 4: **Entity Resolution: clustering results DC (AE, EDESC and SHGP) vs SC (K-means, Birch and DBSCAN) using EmbDi and SBERT on Music Brainz 2K and GeoSet datasets.**

		AE		EDESC		SHGP		K-means		DBSCAN		Birch	
Dataset	**Metric**	EmbDi	SBERT	EmbDi	SBERT	EmbDi	SBERT	EmbDi	SBERT	EmbDi	SBERT	EmbDi	SBERT
	K	684	684	684	668	684	684	684	684	0	1	684	684
Music Brainz	ARI	**0.51**	**0.77**	0.43	0.43	0.20	0.16	0.41	0.38	0.0	0.00	0.41	0.56
	ACC	**0.71**	**0.86**	0.67	0.79	0.51	0.48	0.65	0.67	0.002	0.004	0.67	0.76
	K	786	786	786	786	786	786	786	786	N/A	1	786	688
GeoSet	ARI	**0.61**	**0.85**	0.60	0.81	0.43	0.72	0.57	0.74	0.0	0.0005	0.59	0.31
	ACC	0.72	**0.91**	**0.73**	0.89	0.63	0.84	0.72	0.86	0.001	0.002	0.71	0.59

representation does not perform especially well, but is improved on by AE and EDESC. In EmbDi, high similarity scores may be given even where there are few attributes in common. For example, all rows in the largest cluster contain only the common attribute value *(Language: spa.)*, which occurred frequently. The cosine similarity of EmbDi vectors of two records with different values of *(title, length, artist, album, year)* and the same value of *(Language: spa)* is *0.75*, causing the SC algorithms to cluster them together. The representation learned by AE from EmbDi resolved this issue. (vi) **SC methods were outperformed by DC methods for both datasets.** Although showing some strength, Birch and K-means do not match the feature learning capabilities of DC methods, reflected in the lower ACC scores than AE (0.19 and 0.10 for Music Brainz with SBERT and 0.05 and 0.32 for GeoSet with SBERT, respectively). SHGP, however, failed to produce better clusters than K-means and Birch, with lower ARI scores of 0.14 and 0.16 with EmbDi on GeoSet data. Like schema inference, DBSCAN struggles in entity resolution and produces one cluster due to highly similar dense data regions.

7 DOMAIN DISCOVERY

Domain discovery is the process of identifying collections of values that instantiate an application concept. Discovering domains tends to involve looking for similar collections of values in different dataset columns. Most prior work has used bespoke algorithms [41, 48, 52], but in this section we investigate the use of generic clustering techniques for identifying columns that share domains.

For domain discovery, the clustering problem can be defined as: for a given set of columns $C = \{c_1, c_2, c_3 \ldots c_n\}$ identify every subset $C_s \subseteq C$ that shares a common domain using clustering. To infer a domain from a set of columns, we considered schema-level evidence with pre-trained sentence transformer SBERT and word embedding technique FastText, and Schema+Instance-level with SBERT and EmbDi [14].

We used the Di2KG (Camera and Monitor) datasets[2], which consist of camera and monitor specifications extracted from multiple e-commerce web pages. The datasets are highly heterogeneous in terms of single or multiple sources. For example, synonyms, e.g., *lens* from *www.cambuy.com.au* and *normalized optical zoom* from *buy.net*, semantically represent the same domain. There are several homonyms, i.e., *screen type* is considered in some sources to represent *screen size*. The properties of the datasets evaluated for domain discovery is presented in Table 1. Similar to entity resolution, in some experiments the representation is not well learned in the training of SDCN but by the AE in the pre-training module. Based on the silhouette score, we use the AE instead of SDCN in some domain discovery experiments. The details are given in the hyperparameter setting[3].

We used three embedding methods for column clustering, considering schema-level and schema+instance-level data. To encode column attributes, we used pre-trained models SBERT and FastText as we used in schema inference. To encode columns at schema+instance-level, we utilized the Schema Matching (SM) version of EmbDi (Algorithm 5 in EmbDi [14]) and evaluated skip-gram as a learning method with piece-wise smoothing. In domain discovery, we have a set of columns with cell values that can be represented as a *phrase* in SBERT which is trained on diverse text corpora and can capture semantic and syntactic information. Considering this, we used SBERT to encode column headers and values jointly. SBERT processes each column and generates embeddings representing the semantic content of the column headers and values. Subsequently, the embedding for each column is computed by performing a mean operation on the corresponding column header and value embeddings.

7.1 Results and Discussion

Table 5 shows the clustering results for domain discovery using schema-level data. We observe the following: (i) **All the clustering algorithms perform quite similarly when considering schema-level data**. This suggests that DC is not significantly improving the representation and indicates that the representations used capture the necessary structure and meaningful differences well enough for SC to group suitable column headers, especially in the Camera dataset. (ii) **SHGP outperformed SDCN and EDESC using SBERT with Monitor data**. SHGP obtained an ACC score of 0.03 higher than SDCN and EDESC, in contrast with its performance in domain discovery and entity resolution. SHGP captured more syntactic structures within the column headers and hierarchically divided the graph into various sub-graphs with similar features. For example, attributes *(max resolutions)*, *(resolution)* and *(supported graphics resolutions)* are true positives in GT and SHGP but false negatives in the SDCN prediction. (iii) **SBERT and FastText with schema-level data are much more similar than in schema inference.** SBERT is leading by 0.03 ARI in SDCN, 0.08 ARI score in EDESC, and 0.05 ARI score in SHGP, a relatively small difference compared to the performance of FastText in SI. This is because, in schema inference, we have long contextual phrases compared to domain discovery. The attribute phrases in the Camera dataset are small, and FastText does not need to consider the order of words to embed, leading to good performance.

Table 6 presents the clustering results for domain discovery using schema+instance-level data. We observe the following: (i) **All**

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

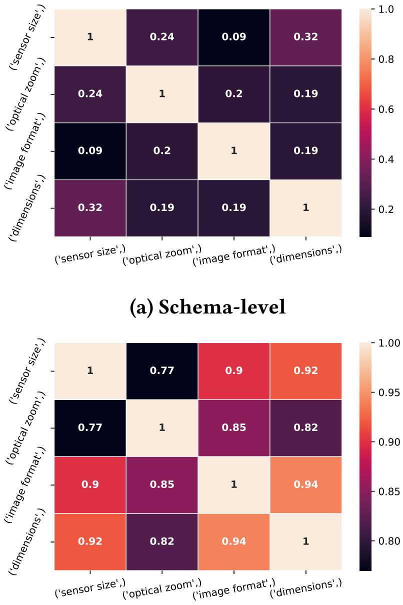
[3] https://github.com/hafizrauf/dc_data-integration

Table 5: Domain discovery: Schema-level clustering resultsDC (SDCN/AE, EDESC and SHGP) vs SC (K-means, Birch and DBSCAN) using SBERT and FastText on Di2KG (Camera and Monitor) datasets.

		SDCN/AE		EDESC		SHGP		K-means		DBSCAN		Birch	
Dataset	**Metric**	SBERT	FastText	SBERT	FastText	SBERT	FastText	SBERT	FastText	SBERT	FastText	SBERT	FastText
	K	42	56	56	56	56	56	56	56	49	47	56	44
Camera	ARI	0.74	**0.71**	**0.78**	0.70	0.66	0.69	0.73	**0.71**	0.73	0.35	0.76	0.58
	ACC	0.69	**0.68**	**0.74**	0.66	0.62	0.65	0.69	0.66	0.69	0.53	0.70	0.62
	K	81	81	81	81	81	81	81	81	99	100	81	81
Monitor	ARI	0.59	**0.57**	0.59	0.57	**0.59**	0.54	0.57	0.55	0.27	0.30	0.52	0.54
	ACC	0.58	**0.56**	0.58	0.54	**0.61**	0.55	0.57	0.54	0.50	0.51	0.54	**0.56**

Table 6: Domain discovery: Schema+Instance-level clustering results DC (SDCN/AE, EDESC and SHGP) vs SC (K-means, Birch and DBSCAN) using SBERT and EmbDi on Di2KG (Camera and Monitor) datasets.

		SDCN/AE		EDESC		SHGP		K-means		DBSCAN		Birch	
Dataset	**Metric**	SBERT	EmbDi	SBERT	EmbDi	SBERT	EmbDi	SBERT	EmbDi	SBERT	EmbDi	SBERT	EmbDi
	K	51	56	56	56	56	56	56	56	42	1	56	56
Camera	ARI	**0.86**	**0.13**	0.81	0.11	0.47	0.07	0.51	0.12	-0.005	0.02	0.78	0.03
	ACC	**0.80**	**0.17**	0.78	0.15	0.56	0.11	0.56	0.15	0.25	0.13	0.74	0.14
	K	81	81	81	81	81	81	81	81	87	2	81	81
Monitor	ARI	**0.64**	**0.06**	0.62	**0.06**	0.51	0.02	0.58	**0.06**	0.05	0.002	0.60	0.04
	ACC	**0.63**	**0.13**	0.62	**0.13**	0.53	0.08	0.58	**0.13**	0.38	0.06	0.61	**0.13**



(a) Schema-level

(b) Schema+instance-level

Figure 5: Heat map representation of SBERT (schema-level) (a) and EmbDi (b) with SDCN on Camera data. When instance-level data is added for encoding, we can observe that all true negative cases in (a) are false positives in (b).

clustering methods struggled to integrate Schema+Instance data with EmbDi and showed much better performance with SBERT on all datasets. EmbDi failed to produce suitable embeddings for column headers and values because EmbDi emphasizes relationships between columns in a table, which are not especially relevant to domain discovery. In contrast, SBERT considers the textual context for each column header and value and then combines them, ignoring surrounding columns. Furthermore, the performance of EmbDi is also impacted by the syntactic dissimilarity between column headers. For example, two-column attributes in Camera data *(image size pixels)* and *(max resolution)* are lexically dissimilar with a cosine similarity of 0; however, they can have similar instance values. The largest cluster predicted by EDESC with EmbDi contains 1151 columns that belong to 13 GT domains but represent one domain in the prediction, which shows a high false positive rate. Some examples of domains clustered by EDESC with EmbDi but not in the GT clusters are *(battery type, lens type, battery life, camera type).* We used heat maps (Figure 5) to analyze how the distance vectors of columns are similar or dissimilar. We investigate how adding instance-level data affects the representation of columns. For heat map visualization, columns are selected randomly from the predicted clusters of SDCN encoded with SBERT (schema-level) and EmbDi (schema+instance-level). Unlike SBERT, EmbDi with SDCN groups those columns that are neither syntactically similar nor belong to the same domain. Adding instance-level data gives rise to a poorer encoding for SDCN with EmbDi. Figure 5a confirms that different columns that should be in the same cluster are in different clusters. This indicates that the column headers are lexically different and suitable for models pre-trained on large dictionaries. Figure 5b shows that all the example columns belong to different real-world domains but are still assigned to one cluster. SBERT with SDCN managed to segregate those columns, which are lexically different, from schema-level evidence. (ii) **Combining instance-level data with schema-level data helps in domain discovery but not schema inference for all clustering methods.** In domain discovery, the column headers have high syntactic similarity despite belonging to different domains. Adding more relevant information from column values into the feature space makes the features more different, and clustering methods find criteria to differentiate between clusters. On the other hand, the table attributes in schema inference have high semantic similarity with more similar table values, making the features more similar to each other. Adding table values into feature space may lead to highly overlapping features, which clustering methods find hard to cluster correctly. For example, in domain discovery, the SBERT cosine similarity between two column headers *(headphone outputs)* and *(headphone out)* that belonging to different domains is 0.78, which is relatively high and they are likely to be placed in one cluster; however, when we add instance-level data *(headphone outputs: 1)* and *(headphone out: yes)*, this provides additional information, making the two features less similar.

8 DISCUSSION

The following are cross-cutting findings from the experiments:

SDCN with SBERT performed well in several problems (particularly for schema inference with schema-level data on web tables, entity resolution on both datasets and domain discovery with schema+instance-level on the Camera dataset) compared to other embedding methods. SDCN allows for fine-tuning of SBERT by way of the lower-dimensional latent space of the AE, potentially capturing deeper semantic relationships in sentences. For example, for schema inference, the representation of the two sets of table attributes *(common name, scientific name, family)* and *(species, scientific name, day, high count, total count)* from table *Bird* are well learned by AE when SBERT is fine-tuned compared to FastText because SBERT considers the context of these attributes, whereas FastText uses sub-word information. The two sets of table attributes are correctly clustered together in SDCN with SBERT but are apart with FastText.

EDESC performed better clustering when there were a large number of clusters with small cluster cardinality (particularly for schema inference with schema-level on web tables using FastText, domain discovery with schema-level on Camera datasets using SBERT and with schema+instance-level on Monitor datasets using EmbDi). When there are a large number of clusters, and each cluster contains a relatively small number of instances, those small clusters tend to contain instances with a higher degree of separation (the similarity between instances is higher within the same cluster and lower between different clusters). This occurrence of clusters with prominent distinct features reduces the overlap between different subspaces. In contrast, where there are a small number of large clusters, these tend to have lower inter-cluster distances, increasing the probability of overlapping subspaces in which instances that should be in different clusters are assigned to the same subspace. For example, the GT's mean cluster cardinality for web tables data is 16.5; EDESC with FastText predicted 14 clusters with cardinality below 16.5 compared to SDCN with FastText, which predicted 9, thereby missing more smaller clusters.

SDCN prioritizes cluster quality over quantity. SDCN forms fewer clusters (observed for schema inference with schema-level using SBERT on web tables data and domain discovery schema+instance-level using SBERT on Camera data), but these clusters are denser and better separated than in SC methods, which even when they produce the same number of clusters as in the GT these are less dense and compact. An example of this phenomenon in domain discovery on schema+instance-level Camera data is that SDCN with SBERT formed 42 clusters against 56 GT clusters and yet outperformed Birch and K-means by 0.08 and 0.35 ARI respectively, even though they produced the correct number of clusters.

SHGP performs poorly for all problems except when applied to the Monitor dataset, executing domain discovery on schema-level data using SBERT. Since SHGP uses K-means to cluster the embeddings learned by two modules, *Att-LPA* and *Att-HGNN* (referred to SHGP in Section 3), this indicates that SBERT embeddings of raw columns are more robust than SHGP embeddings (a fine-tuned version of SBERT using *Att-LPA* and *Att-HGNN*).

DBSCAN performed poorly for all experiments in schema inference and entity resolution and predicted a minimal number of clusters, sometimes a singular cluster. We observed that DBSCAN tends to merge distinct clusters into one cluster because all clusters have similar densities. In DBSCAN, a cluster is a dense space region separated by lower-density regions. If all the instances fall in the same density region, it becomes difficult for DBSCAN to differentiate between clusters. We validate this observation using the Kolmogorov-Smirnov (KS) test [3, 59] to determine the similarity in density distributions between different features. The KS test compares the cumulative distributions of the pairs of instances to determine their differences. The null hypothesis is that the pairs of instances are drawn from the same distribution. We applied the KS test pairwise to all possible pairs of features obtained from SBERT embeddings of web tables data for schema inference considering schema-level data. The KS test returns two measures, the K-statistic (smaller value indicates that all features share the same distribution or density) and p-value (smaller value suggests rejecting the null hypothesis). We obtained mean K-statistic = 0.06, indicating that all features represent the same distribution (similar densities), and mean p-value = 0.65, confirming that we cannot reject the null hypothesis.

9 COMPARISON WITH BESPOKE METHODS

Sections 5 to 7 compare DC and SC algorithms for data integration tasks, showing that DC can provide significant benefits over SC for these tasks. This section investigates how DC performs compared with state-of-the-art unsupervised approaches to *schema inference*, *entity resolution* and *domain discovery*. Throughout, we compare the bespoke methods with SDCN, using the most effective representation found for each problem in Sections 5 to 7.

Schema Inference. There are few works on schema inference for tables that can be seen as directly competing with the clustering approach investigated in Section 5. The first approach we compare with is explicitly associated with a schema inference proposal [1], and uses D^3L [10], which combines several LSH-indexes to measure the similarity of datasets, along with an SC algorithm. The second approach we compare with, Starmie [25], uses contrastive learning over the ROBERTa language model [42] to generate fine-tuned representations for table and column similarity, and proposes clustering results over these representations using an SC algorithm. Thus, as a baseline, this combines SC with a language model that has been fine-tuned for dataset similarity. The results are presented in Table 7. It can be observed that the DC results are better than Starmie for both datasets and D^3L for the TUS dataset. D^3L uses an SC algorithm to perform clustering over the similarity matrix. When we use AE over the D^3L similarity matrix, AE obtained better results than D^3L (0.60 ARI and 0.73 ACC). This indicates that D^3L outperforms Tabnet (in terms of better representation), and AE outperforms D^3L (in terms of better clustering). In D^3L, the comparison criteria are mostly syntactic, but include features of both headers and instances. In Starmie, the representation is learned from instances, and has been fine-tuned for similarity comparison, whereas in

Table 7: Comparing DC with bespoke solutions.

	Schema Inference						Entity Resolution					Domain Discovery			
dataset	TUS		web tables				GeoSet		Music Brainz			Camera		Monitor	
encoding	Tabnet		Tabnet				SBERT		SBERT			SBERT		SBERT	
metric	ARI	ACC	ARI	ACC			ARI	ACC	ARI	ACC		ARI	ACC	ARI	ACC
D^3L	**0.56**	**0.72**	0.14	0.31		Jaccard	0.07	0.58	0.16	0.59	D4	0.29	0.27	N/A	N/A
Starmie	0.11	0.33	0.10	0.31	JedAI	Cosine	0.32	0.64	0.58	0.69	Starmie	-0.007	0.14	0.001	0.09
SDCN	0.34	0.45	**0.45**	**0.55**		Dice	0.31	0.64	0.57	0.69	**SDCN**	**0.86**	**0.80**	**0.64**	**0.63**
					SDCN		**0.85**	**0.91**	**0.77**	**0.86**					

SDCN the representation has been fine-tuned for similarity clustering. Starmie uses a connected component algorithm [25] to produce clusters, and we observe that Starmie encountered an over-segmentation issue where it produces too many small clusters (173 with 113 unary clusters) against 26 GT clusters on web tables data. This indicates that the similarity graph obtained from Starmie column embeddings given to the connected component algorithm is not robust enough, making it sensitive to minor variations.

Entity Resolution. There are many results on entity resolution (ER); as a comparator, we use a workflow from JedAI [44, 49]. JedAI is a platform that brings together state-of-the-art clustering algorithms to support empirical evaluation. We use the schema-agnostic workflow from JedAI because: (i) it is unsupervised and the deep clustering approaches are also schema-agnostic; (ii) the default parameters of the workflow have been derived from experience with many datasets and thus should provide robust performance; (iii) different similarity metrics are supported; and (iv) the workflow includes a clustering step, whereas quite a lot of ER proposals stop at pairwise comparison. The results are presented in Table 7. It can be seen that SDCN outperforms the JedAI unsupervised workflow with all similarity measures; the issue with the performance of JedAI for these datasets is primarily during the clustering stage, with too many small clusters being produced.

Domain Discovery. There are not many fully unsupervised domain discovery proposals. We experiment with: D4 (Data Driven Domain Discovery) [48], a bespoke algorithm that seeks to identify domains from overlaps in column extents; and Starmie [25], which as discussed for Schema Inference, uses self-supervised contrastive learning over the ROBERTa language model to produce column embeddings. We note that D4 may infer that a column participates in several domains, whereas by clustering columns, we have been associating each column with a single domain. To overcome this issue for the experiments, we associated each column with the D4 domain that has the greatest coverage. The results are presented in Table 7. The following can be observed: (i) DC outperforms D4 on the benchmark dataset; this is because D4 assumes consistent representations for column values, and the datasets used regularly manifest representational inconsistencies. Although we have successfully run D4 on several datasets, it did not manage to identify domains in the Monitor dataset[4]. (ii) Starmie has not performed well for this task. Starmie's performance heavily relies on the fine-tuning process over ROBERTa, which is less specialized in computing contextual similarities at the sentence level than SBERT. The embeddings produced by Starmie were observed to have high intra-class variability (variations within the same cluster) compared to those obtained by SBERT, which directly affects the clustering performance.

10 CONCLUSIONS

We have investigated the application of DC for *schema inference*, *entity resolution* and *domain discovery*, tasks that cluster tables, rows and columns, respectively. Experiments have explored the use of DC algorithms on these mainstream data management tasks, using a variety of embeddings for complete tables, columns, and rows. Results have been reported comparing three existing DC algorithms with three non-DC algorithms representing different clustering paradigms. The results show that DC algorithms consistently outperform non-DC clustering algorithms for data integration tasks, thus motivating their adoption to cluster tabular datasets or their components. We have also compared the DC proposals with state-of-the-art algorithms for each of *schema inference*, *entity resolution* and *domain discovery*, where once again the results are encouraging, consistently outperforming the bespoke proposals.

We identified potential future research opportunities by empirical evaluation, which include (i) Exploring distance functions to effectively measure row-to-row, column-to-column and table-to-table similarity in the latent space for deep clustering. (ii) Efficient transformation of dense to sparse matrices before learning the representation. (iii) Exploring different techniques to minimize the effect of large numbers of clusters on deep clustering performance; as the number of clusters grows, the model's complexity increases, and it becomes more likely that some clusters will be very similar, leading to more challenging optimization problems.

ACKNOWLEDGMENTS

The authors would like to acknowledge the assistance given by Research IT and the use of the Computational Shared Facility (CSF) at The University of Manchester. Hafiz Tayyab Rauf is supported by a studentship from The University of Manchester.

REFERENCES

[1] Nour Alhammad, Alex Bogatu, and Norman W. Paton. 2022. Towards Schema Inference for Data Lakes. *CoRR* abs/2206.03881 (2022). https://doi.org/10.48550/ARXIV.2206.03881 arXiv:2206.03881

[2] Elie Aljalbout, Vladimir Golkov, Yawar Siddiqui, and Daniel Cremers. 2018. Clustering with Deep Learning: Taxonomy and New Methods. *CoRR* abs/1801.07648 (2018). arXiv:1801.07648 http://arxiv.org/abs/1801.07648

[3] KOLMOGOROV AN. 1933. Sulla determinazione empirica di una legge didistribuzione. *Giorn Dell'inst Ital Degli Att* 4 (1933), 89–91.

[4] Sercan Ö. Arik and Tomas Pfister. 2021. TabNet: Attentive Interpretable Tabular Learning. 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, 6679–6687. https://ojs.aaai.org/index.php/AAAI/article/view/16826

[4] Specifically, D4 returned context signature count 0 with the Monitor dataset

[5] 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. https://doi.org/10.1007/s00778-018-0532-7

[6] Gilbert Badaro and Paolo Papotti. 2022. Transformers for Tabular Data Representation: A Tutorial on Models and Applications. *Proc. VLDB Endow.* 15, 12 (2022), 3746–3749. https://www.vldb.org/pvldb/vol15/p3746-badaro.pdf

[7] Jinqiang Bai, Shiguo Lian, Zhaoxiang Liu, Kai Wang, and Dijun Liu. 2018. Deep Learning Based Robot for Automatically Picking Up Garbage on the Grass. *IEEE Trans. Consumer Electron.* 64, 3 (2018), 382–389. https://doi.org/10.1109/TCE.2018.2859629

[8] 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.

[9] 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. https://doi.org/10.1145/3366423.3380214

[10] 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.* 709–720. https://doi.org/10.1109/ICDE48307.2020.00067

[11] Alex Bogatu, Norman W. Paton, Mark Douthwaite, Stuart Davie, and André Freitas. 2021. Cost-effective Variational Active Entity Resolution. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021.* 1272–1283. https://doi.org/10.1109/ICDE51399.2021.00114

[12] Deng Cai, Xiaofei He, and Jiawei Han. 2005. Document Clustering Using Locality Preserving Indexing. *IEEE Trans. Knowl. Data Eng.* 17, 12 (2005), 1624–1637. https://doi.org/10.1109/TKDE.2005.198

[13] 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. https://doi.org/10.1109/CVPR52688.2022.00012

[14] 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. https://doi.org/10.1145/3318464.3389742

[15] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. 2018. Deep Clustering for Unsupervised Learning of Visual Features. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIV.* 139–156. https://doi.org/10.1007/978-3-030-01264-9_9

[16] 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. https://doi.org/10.1007/s00778-018-0528-3

[17] 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. https://doi.org/10.1145/3418896

[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. https://doi.org/10.1007/s10618-009-0155-0

[19] Zhiyuan Dang, Cheng Deng, Xu Yang, Kun Wei, and Heng Huang. 2021. Nearest Neighbor Matching for Deep Clustering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021.* Computer Vision Foundation / IEEE, 13693–13702. https://doi.org/10.1109/CVPR46437.2021.01348

[20] Daniel P. M. de Mello, Renato M. Assunção, and Fabricio Murai. 2022. Top-Down Deep Clustering with Multi-Generator GANs. 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, 7770–7778. https://ojs.aaai.org/index.php/AAAI/article/view/20745

[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. https://doi.org/10.1109/ICCV.2017.612

[22] 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. https://doi.org/10.14778/3236187.3236198

[23] 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. https://doi.org/10.1109/TKDE.2007.250581

[24] 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

[25] 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. https://doi.org/10.14778/3587136.3587146

[26] Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. 2020. SCAN: Learning to Classify Images Without Labels. In *Computer Vision - ECCV 2020 - 16th European Conference Proceedings, Part X (LNCS, Vol. 12355)*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer, 268–285. https://doi.org/10.1007/978-3-030-58607-2_16

[27] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. 2021. Revisiting Deep Learning Models for Tabular Data. 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.). 18932–18943. https://proceedings.neurips.cc/paper/2021/hash/9d86d83f925f2149e9edb0ac3b49229c-Abstract.html

[28] 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

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

[30] 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. https://doi.org/10.14778/1687627.1687771

[31] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *science* 313, 5786 (2006), 504–507.

[32] 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

[33] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and João Carreira. 2021. Perceiver: General Perception with Iterative Attention. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 4651–4664. http://proceedings.mlr.press/v139/jaegle21a.html

[34] Pan Ji, Tong Zhang, Hongdong Li, Mathieu Salzmann, and Ian D. Reid. 2017. Deep Subspace Clustering Networks. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 24–33. https://proceedings.neurips.cc/paper/2017/hash/e369853df766fa44e1ed0ff613f563bd-Abstract.html

[35] Licheng Jiao and Jin Zhao. 2019. A Survey on the New Generation of Deep Learning in Image Processing. *IEEE Access* 7 (2019), 172231–172263. https://doi.org/10.1109/ACCESS.2019.2956508

[36] 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.

[37] Young Jin Kim and Hany Hassan. 2020. FastFormers: Highly Efficient Transformer Models for Natural Language Understanding. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing, SustaiNLP@EMNLP 2020, Online, November 20, 2020*, Nafise Sadat Moosavi, Angela Fan, Vered Shwartz, Goran Glavas, Shafiq R. Joty, Alex Wang, and Thomas Wolf (Eds.). Association for Computational Linguistics, 149–158. https://doi.org/10.18653/v1/2020.sustainlp-1.20

[38] Jakub Klímek and Martin Necaský. 2010. Reverse-engineering of XML Schemas: A Survey. In *Proceedings of the Dateso 2010 Annual International Workshop on DAtabases, TExts, Specifications and Objects, Stedronin-Plazy, Czech Republic, April 21-23, 2010.* 96–107. http://ceur-ws.org/Vol-567/paper19.pdf

[39] Sampo Kuutti, Richard Bowden, Yaochu Jin, Phil Barber, and Saber Fallah. 2021. A Survey of Deep Learning Applications to Autonomous Vehicle Control. *IEEE Trans. Intell. Transp. Syst.* 22, 2 (2021), 712–733. https://doi.org/10.1109/TITS.2019.2962338

[40] Ivano Lauriola, Alberto Lavelli, and Fabio Aiolli. 2022. An introduction to Deep Learning in Natural Language Processing: Models, techniques, and tools. *Neurocomputing* 470 (2022), 443–456. https://doi.org/10.1016/j.neucom.2021.05.103

[41] 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.* 1873–1882. https://doi.org/10.1145/3097983.3098102

[42] 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

[43] 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. https://ojs.aaai.org/index.php/AAAI/article/view/20726

[44] 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. https://doi.org/10.1016/J.IS.2021.101830

[45] Leland McInnes and John Healy. 2018. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *CoRR* abs/1802.03426 (2018). arXiv:1802.03426 http://arxiv.org/abs/1802.03426

[46] Erxue Min, Xifeng Guo, Qiang Liu, Gen Zhang, Jianjing Cui, and Jun Long. 2018. A Survey of Clustering With Deep Learning: From the Perspective of Network Architecture. *IEEE Access* 6 (2018), 39501–39514. https://doi.org/10.1109/ACCESS.2018.2855437

[47] 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. https://doi.org/10.14778/3192965.3192973

[48] 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. https://doi.org/10.14778/3384345.3384346

[49] 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. https://doi.org/10.1016/J.IS.2020.101565

[50] Sungwon Park, Sungwon Han, Sundong Kim, Danu Kim, Sungkyu Park, Seunghoon Hong, and Meeyoung Cha. 2021. Improving Unsupervised Image Clustering With Robust Learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021.* Computer Vision Foundation / IEEE, 12278–12287. https://doi.org/10.1109/CVPR46437.2021.01210

[51] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[52] Federico Pial, Paolo Atzeni, Paolo Merialdo, and Divesh Srivastava. 2022. Fine-grained semantic type discovery for heterogeneous sources using clustering. *VLDB Journal* (2022). https://doi.org/10.1007/s00778-022-00743-3

[53] Stanislav Pidhorskyi, Ranya Almohsen, and Gianfranco Doretto. 2018. Generative Probabilistic Novelty Detection with Adversarial Autoencoders. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (Eds.). 6823–6834. https://proceedings.neurips.cc/paper/2018/hash/5421e013565f7f1afa0cfe8ad87a99ab-Abstract.html

[54] 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. https://doi.org/10.18653/v1/d19-1410

[55] 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. https://doi.org/10.5441/002/edbt.2017.20

[56] Peter J Rousseeuw. 1987. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics* 20 (1987), 53–65.

[57] Alieh Saeedi, Eric Peukert, and Erhard Rahm. 2017. Comparative Evaluation of Distributed Clustering Schemes for Multi-source Entity Resolution. In *Advances in Databases and Information Systems - 21st European Conference, ADBIS 2017, Nicosia, Cyprus, September 24-27, 2017, Proceedings (Lecture Notes in Computer Science, Vol. 10509)*, Marite Kirikova, Kjetil Nørvåg, and George A. Papadopoulos (Eds.). Springer, 278–293. https://doi.org/10.1007/978-3-319-66917-5_19

[58] Erich Schubert, Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. 2017. DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN. *ACM Trans. Database Syst.* 42, 3 (2017), 19:1–19:21. https://doi.org/10.1145/3068335

[59] Nickolay Smirnov. 1948. Table for estimating the goodness of fit of empirical distributions. *The annals of mathematical statistics* 19, 2 (1948), 279–281.

[60] Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C. Bayan Bruss, and Tom Goldstein. 2021. SAINT: Improved Neural Networks for Tabular Data via Row Attention and Contrastive Pre-Training. *CoRR* abs/2106.01342 (2021). arXiv:2106.01342 https://arxiv.org/abs/2106.01342

[61] Chunfeng Song, Feng Liu, Yongzhen Huang, Liang Wang, and Tieniu Tan. 2013. Auto-encoder Based Data Clustering. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications - 18th Iberoamerican Congress, CIARP 2013, Havana, Cuba, November 20-23, 2013, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 8258)*, José Ruiz-Shulcloper and Gabriella Sanniti di Baja (Eds.). Springer, 117–124. https://doi.org/10.1007/978-3-642-41822-8_15

[62] Saravanan Thirumuruganathan, Han Li, Nan Tang, Mourad Ouzzani, Yash Govind, Derek Paulsen, Glenn Fung, and AnHai Doan. 2021. Deep Learning for Blocking in Entity Matching: A Design Space Exploration. *Proc. VLDB Endow.* 14, 11 (2021), 2459–2472. https://doi.org/10.14778/3476249.3476294

[63] Saravanan Thirumuruganathan, Nan Tang, Mourad Ouzzani, and AnHai Doan. 2020. Data Curation with Deep Learning. In *Proceedings of the 23rd International Conference on Extending Database Technology, EDBT 2020, Copenhagen, Denmark, March 30 - April 02, 2020.* 277–286. https://doi.org/10.5441/002/edbt.2020.25

[64] Tsung Wei Tsai, Chongxuan Li, and Jun Zhu. 2021. MiCE: Mixture of Contrastive Experts for Unsupervised Image Clustering. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021.* OpenReview.net. https://openreview.net/forum?id=gV3wdEOGy_V

[65] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.). 5998–6008. https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[66] 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. https://doi.org/10.1109/ICCV.2019.00824

[67] Yifan Xing, Tong He, Tianjun Xiao, Yongxin Wang, Yuanjun Xiong, Wei Xia, David Wipf, Zheng Zhang, and Stefano Soatto. 2021. Learning Hierarchical Graph Neural Networks for Image Clustering. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021.* IEEE, 3447–3457. https://doi.org/10.1109/ICCV48922.2021.00345

[68] Weidi Xu, Haoze Sun, Chao Deng, and Ying Tan. 2017. Variational Autoencoder for Semi-Supervised Text Classification. 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, 3358–3364. http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14299

[69] 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

[70] 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. https://doi.org/10.1109/TIP.2010.2049235

[71] Fei Ye and Adrian G. Bors. 2022. Deep Mixture Generative Autoencoders. *IEEE Trans. Neural Networks Learn. Syst.* 33, 10 (2022), 5789–5803. https://doi.org/10.1109/TNNLS.2021.3071401

[72] 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.* 319–330. http://dl.acm.org/citation.cfm?id=1164156

[73] Junjian Zhang, Chun-Guang Li, Chong You, Xianbiao Qi, Honggang Zhang, Jun Guo, and Zhouchen Lin. 2019. Self-Supervised Convolutional Subspace Clustering Network. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019.* Computer Vision Foundation / IEEE, 5473–5482. https://doi.org/10.1109/CVPR.2019.00562

[74] 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. https://doi.org/10.1145/233269.233324

[75] Huasong Zhong, Jianlong Wu, Chong Chen, Jianqiang Huang, Minghua Deng, Liqiang Nie, Zhouchen Lin, and Xian-Sheng Hua. 2021. Graph Contrastive Clustering. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021.* IEEE, 9204–9213. https://doi.org/10.1109/ICCV48922.2021.00909

[76] Pan Zhou, Yunqing Hou, and Jiashi Feng. 2018. Deep Adversarial Subspace Clustering. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018.* Computer Vision Foundation / IEEE Computer Society, 1596–1604. https://doi.org/10.1109/CVPR.2018.00172

[77] 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). https://doi.org/10.48550/arXiv.2206.07579 arXiv:2206.07579