Scalable Tabular Hierarchical Metadata Classification in Heterogeneous Structured Large-scale Datasets using Contrastive Learning

Bhimesh Kandibedala*, Gyanendra Shrestha*
Florida State University
Tallahassee, USA

Anna Pyayt
University of South Florida
Tampa, USA

Todor Ivanov, Michael Gubanov
Florida State University
Tallahassee, USA

***Abstract*—Tabular metadata (i.e., attributes in a table) identification and classification is a *fundamental* problem in large-scale data management of structured corpora, especially for complex tables rich in multi-level *hierarchical* metadata with nesting. Medical, security, data science research literature, Web tables, contain thousands of such complex tables, but often lack or incorrectly label their complex metadata. In this work, we describe an unsupervised, scalable, contrastive-learning approach for classification of multi-layer, hierarchical metadata in such tables. We compared it to the state of the art (SOTA) as well as the latest Large Language Models (LLMs), such as OpenAI GPT 3.5 and 4 with and without Retrieval Augmented Generation (RAG) on several large-scale heterogeneous datasets. We outperform SOTA and LLMs in classifying horizontal metadata (HMD) of deep levels (3-5) and for all levels (1-3) of vertical metadata (VMD). For HMD levels 1-2, SOTA outperforms us insignificantly, with a delta of ≈1%. LLMs with/without RAG slightly outperform us with deltas of 4-5% in accuracy for HMD level 1, but we significantly outperformed LLMs/LLMs+RAG with delta up to 29% for all other levels 2-5 HMD and up to 87% delta for VMD.**

***Index Terms*—Embeddings, Contrastive learning, Large-scale Data management, Large Language Models**

I. Introduction

Large-scale, heterogeneous datasets exist in a variety of domains - on the Web, in medical research, education, government, etc. Some of such popular datasets are the Web Data Commons (WDC) [1], the COVID-19 Open Research Dataset (CORD-19) [2], and the Census Bureau [3]. Such corpora exhibit a wealth of useful, differently structured data originating from millions of sources. The variety of schemas is observed not only among different sources, but also even among tables on the same topic. For instance, two tables on the same topic - e.g. Songs, Vaccine side effects, or Cancer therapy efficacy, can have significantly different schemas in different sources. This variability makes accessing structured data at scale a very time consuming, labor-intensive process [4]–[11]. Additionally, some sources contain mostly *relational* tables, whereas others, especially in domains such as medical, scientific, finance, often employ more complex tables, featuring hierarchical vertical and/or horizontal metadata with nesting [12]. It introduces significant complexity in managing and accessing them and calls for advanced techniques for effective identification of their complex metadata [13]–[17].

Metadata, in this context, refers to a set of attributes that constitute the table's schema (see Figure 1, marked as *Metadata*) that define both its structure and content. These attributes can be categorized into column-level (horizontal) and/or row-level (vertical) metadata. In complex tables, both vertical metadata (VMD) and horizontal metadata (HMD) are present. Figure 1 (a) illustrates an example of such table having 1 level of HMD and 3 levels of VMD, both *hierarchical*, i.e. consist of multiple rows/levels corresponding to subcategories. For example, the data value "14,373" corresponds to the "student enrollment" at "Binghamton" in the "State University of New York". To fully comprehend semantics of this value, both HMD ("*Student enrollment*") and VMD ("*New York*" → "*State University of New York*" → "*Binghamton*") have to be recognized and interpreted as *Metadata* correctly. Similarly, "New York" in Figure 1(a) is not merely a data value, but a hierarchical VMD element having children/subcategories (*Universities* and *Cities* in this case). VMD elements in a table sometimes may have blanks cells in the preceding top row (in the 1st row in this case). Such metadata, if interpreted in isolation from the remaining table content, can be easily misread as just a table cell value in a tuple. However, in this table, "New York" serves as metadata (i.e. VMD attribute name) rather than a basic data value, categorizing multiple institutions and their respective cities in a hierarchy (columns 2 and 3 in Figure 1(a)). This semantic distinction is crucial - failing to recognize it as metadata can lead to a partial semantics loss of the remaining data in the table. Without identifying the hierarchical structure of VMD, tuples lose their contextual associations with each VMD. In Figure 1(a), tuples 2-28 fall under a three-level hierarchical VMD structure; VMD level 1 ("New York") appears only in row 2, while rows 3-28 have blank cells in this column. For example, if we consider row 10 and mistreat VMD as just data, we would lose the fact that "Stony Brook" belongs to "State University of New York", which in turn belongs to "New York". Similarly, VMD level 2 ("State University of New York") spans rows 4-10, but the attribute name is present only in row 4. If we extract row 10 and treat the hierarchical VMD as data, then it would appear as just an independent entity rather than as a part of

*Equal Contribution.

the "State University of New York". In other words, these hierarchical relationships would be lost. Misinterpretation of VMD and/or HMD as data would make the remaining data rows appear disconnected from an essential part of their semantics; the key contextual information–such as university affiliations or locations for most of them would be lost. Therefore, recognizing and correctly classifying these hierarchical metadata structures is key for correct data comprehension and multiple downstream tasks in data analysis, table parsing, and information extraction.

Due to the differences in structure and metadata variety, accurate and scalable metadata annotation has emerged as a hot topic of ongoing research [18]–[22]. Although state of the art (SOTA) shows promising performance, the datasets used for evaluation often comprise a relatively small number of sources and are thus quite *homogeneous* - i.e. do not exhibit a substantial variety of metadata found in large-scale datasets composed of thousands of sources, which are truly heterogeneous. The authors of each source have the liberty to design their schemas, which leads to having a significant variability in tables' structure and metadata naming convention. Consequently, an algorithm or model that fits one source often does not perform that well on other sources unless the schemas are similar. To demonstrate that an approach scales up to many diverse sources, the evaluation datasets should include a variety of different heterogeneous sources [4], [6], [8]–[11], [23], [24].

Here, we introduce a novel, unsupervised, scalable contrastive-learning approach, suitable for classification and annotation of metadata both in relational tables and tables with hierarchical metadata and nesting [12], [25]–[31]. We construct the *centroid* embedding vectors for tabular metadata and data in the corpus and then calculate a series of angles that allow us to annotate complex multi-level horizontal or vertical metadata with high accuracy. To gauge scalability and generality, we have evaluated it on 6 large-scale structured datasets against SOTA [21], [22], [32] and 2 major LLMs with and without Retrieval Augmented Generation (RAG) [33].

Our approach performs high accuracy VMD classification (up to 96%) compared to 90.4% maximum of SOTA for 1st and 2nd levels VMD combined and outperforms SOTA on the remaining, deeper VMD levels. We outperform SOTA on the HMD levels 2-5. LLMs without RAG slightly outperform us with a delta of 4% in accuracy for HMD level 1, but we outperformed LLMs with delta of 29% for levels 2-5 HMD and up to 87% delta for VMD. Likewise, LLMs+RAG slightly outperform us on HMD level 1 with a delta in accuracy of 5%, while we outperform them with deltas up to 12% for HMD levels 2-5, and up to 15% for VMD.

The rest of the paper is structured as follows. First, we define the terminology that we used throughout the paper. Then we describe the methodology, followed by the experimental study and evaluation. We finish with the related work and conclusion.

II. Preliminaries

Definitions 1, 2, and 4 have been taken verbatim from our group's publication [34]. *Relational tables* [35], have the following properties: values are atomic, each column has values of the same type, and each column has a unique label (i.e., attribute name). The set of all attribute names is called table schema or metadata.

Definition 1 (Metadata). A set of *attributes* or domains of a table, which constitutes a part of the table's schema [36]. Metadata can take a row/several rows (e.g., {"student enrollment",..., "Total civilians"} in Figure 1(a) marked as *Metadata Lev.1 HMD*) or a column/columns (e.g., {"NEW YORK", "INDIANA"} in Figure 1(a) marked as *Lev.1 VMD*).

Definition 2 (Cell). A value found at the intersection of a row and a column in a table (e.g., "61" in Figure 1(a) corresponding to row 2 and column 5, is the cell value). A *relational* table has C*R cells total, where C is the number of columns and R is the number of rows.

Definition 3 (Level). A row number counted from the very first table row (that usually contains attributes). In turn, $M_{xa_1}...M_{xa_n}$ are called *metadata* levels if an entire row consists of metadata, and $D_{xa_1}...D_{xa_n}$ as *data* levels if an entire row consists of data, where x and a_i denote the level and the cell's index respectively. For example, in Figure 1(b), rows 1-5, numbered in red, constitute 5 *metadata* levels marked as *Metadata Lev.1-5 HMD*, and the remaining rows represent *data* levels. The same applies to columns, where the first two columns with metadata, labeled in green, represent 2 *metadata* levels marked as *Metadata Lev.1-2 VMD*, while the remaining columns are *data* levels.

Definition 4 (Generally structured table). A table, which metadata can be found not only in the top row as in a relational table [36], but also in a column (usually leftmost, see Figure 1(a), and (b)). Unlike in a relational table, metadata may also span several rows or columns and exhibit hierarchy (see Figure 1(b), and (c)). We refer to rows with metadata - *horizontal metadata (HMD)* and denote them as MH_{xy}, where MH is short for "MetadataHorizontal", x and y represent the row/column indices respectively. We refer to columns with metadata - *vertical metadata (VMD)* and denote them as MV_{xy}, where MV is short for "MetadataVertical", x and y represent the row/column indices respectively. A *generally structured* table may have more than one horizontal and vertical metadata row/column. Furthermore, metadata may also exist in the middle of the table, called *central horizontal metadata (CMD)* and denoted MCH_{xy}, where MCH is short for "MetadataCentral", x and y represent the row/column indices respectively. All data cells in the table are denoted as D_{xy}, where D is short for "Data", x and y correspond to the row/column indices respectively. We also further refer to such *generally structured* tables with hierarchical metadata as *non-relational* tables for brevity. See Figure 1 for several examples of such tables.

Given a heterogeneous, large-scale dataset consisting of generally structured tables (GST) $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$, our

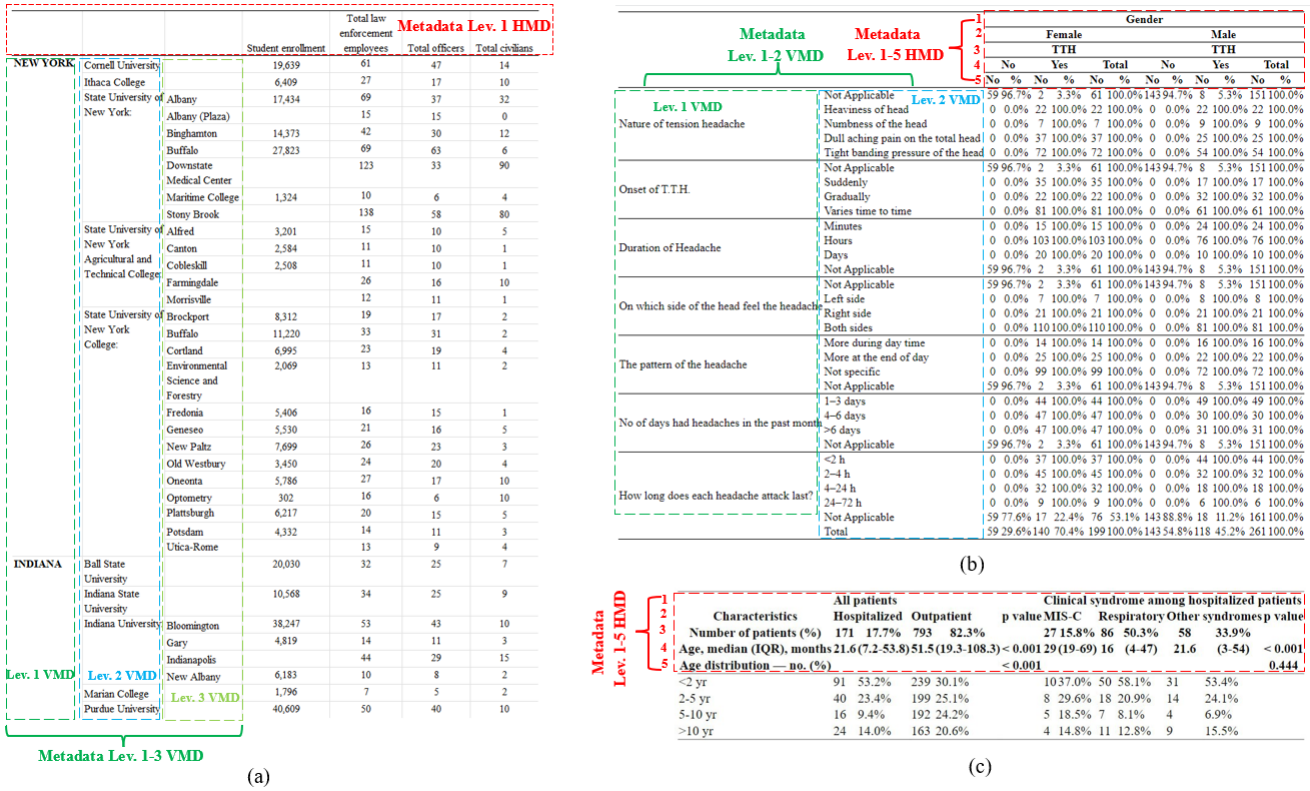

Fig. 1: Tables with Multi-level Hierarchical Metadata. (a) A table with 1 level of Horizontal Metadata (HMD) and 3 levels of Vertical Metadata (VMD). (b) A table with 5 levels of HMD and 2 levels of VMD. (c) A table with 5 levels of HMD only.

objective is to effectively classify and annotate hierarchical metadata for each table T_i. For this we learn a function $f : T_i \rightarrow \{HMD, VMD, D\}$ that, given a table T_i, classifies each table's level (row or column) as HMD, VMD, or otherwise as data. We employ a *contrastive* learning approach, where the embedding vectors are generated for both the metadata and data parts and further used as a basis for the similarity measures.

Formally, we define the metadata classification problem as:

$$\hat{y}_l = f(Emb(T_i), l) \tag{1}$$

where $\hat{y}_l$ is the predicted label (HMD, VMD or D) for level l, and $Emb(T_i)$ is the learned embedding representation of table T_i. To optimize our model, we employ a contrastive loss function based on a Siamese network [37].

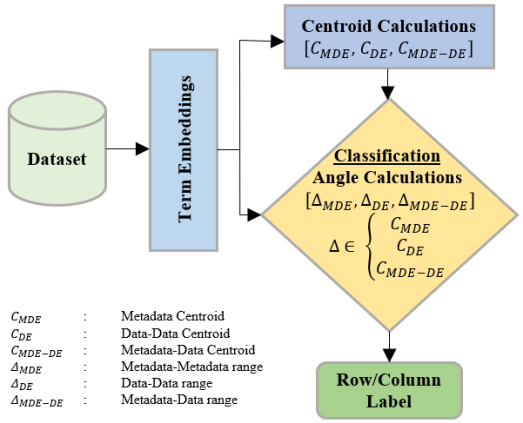

Fig. 2: Contrastive-learning Metadata Classification.

III. Methodology

Reliance only on SOTA supervised learning methods for tabular metadata annotation necessitates large volumes of labeled training data, getting which is expensive, labor intensive, and prone to inconsistencies, such as non-standard or erroneous labels.

To mitigate these concerns, we designed an unsupervised *contrastive-learning* method, aiming to reduce dependency on manual annotation and limitations [38]–[42]. It serves to identify both similarities and dissimilarities between data points (i.e. vectors corresponding to metadata and data in our case) by comparing (*contrasting*) them to each other.

In our approach, we first, generate normalized vectors for terms in a table using the pre-trained embeddings. Next, the method involves identifying gaps in the angular distance between the vectors corresponding to the data and metadata rows. By doing so, we establish a relative measure of semantic similarity, which can be used for accurate metadata identification.

DEFINITION 5 (TERM EMBEDDING). *A high-dimensional vector representations of an individual term in our dataset [43]–[45]. Each term is mapped to a vector in the corresponding vector space, where proximity reflects the semantic similarity between terms. As such, these embedding vectors capture semantic similarity between terms learned from their co-occurrence in different contexts. We use term embeddings to represent terms from tables. For example, "Indiana" in Figure 1(a) will have a corresponding term embedding vector.*

DEFINITION 6 (CENTROID). *The arithmetic mean for a set of vectors. For example, the centroid for metadata labeled as "Metadata Lev. 1 HMD" in Figure 1(a) is obtained by calculating a summation of the embedding vectors corresponding to all attributes marked with this label and dividing by the number of attributes.*

DEFINITION 7 (DEPTH). *Number of levels or rows in a table (e.g., the depth of the table in Figure 1(c) is 9). Depth can also be calculated separately for metadata, e.g. the HMD depth is 5 in Figure 1(c).*

In the following sections, we delve deeper into the specifics of our methodology, depicted in Figure 2. It enhances our ability to understand heterogeneous, structured data better in unsupervised manner. We compute term embeddings by fine-tuning Word2Vec [46] and BioBERT embeddings [47], followed by the calculation of centroids which capture the characteristics of metadata and data. These centroids are utilized in the classification phase. Angles between these embeddings and the precomputed centroids are calculated and compared with the specified ranges to classify each row or column as data or metadata. By iterating over each row and column in the classification phase, our algorithm not only efficiently and accurately labels metadata in an unsupervised manner, but also identifies its depth (the level) in a table.

A. Term Embeddings

The first step of our contrastive learning methodology is the creation of term embeddings, a process integral to the semantic analysis and representation of our tabular data.

Term embeddings provide a numerical representation of the terms in our dataset, encapsulating their semantic relationships and linguistic context in a high-dimensional vector space. We generate these embeddings using Word2Vec and BioBERT models [44], [46], [47] that we fine-tune on our datasets. See Figure 3 for a more detailed logic of this process. BioBERT is a SOTA transformer-based model pre-trained on large biomedical corpora, enabling it to capture the nuances and terminologies of the biomedical domain more effectively than other transformer models. Word2Vec is a popular classic embedding model that is significantly faster to train and use compared to transformer-based models, making it a practical choice for generating term embeddings when computational resources or time are limited. The combination of BioBERT and Word2Vec allows us to leverage the strengths of both models: the domain-specific of BioBERT and the computational efficiency of Word2Vec.

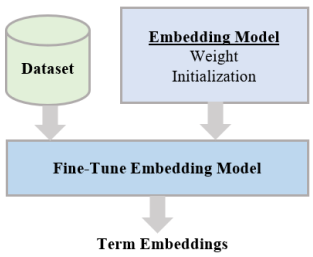

Fig. 3: The Workflow for Generating Term Embeddings.

B. Calculating Centroid Vectors

DEFINITION 8 (AGGREGATED LEVEL). *A summation of the embedding vectors [48], [49] corresponding to all terms in each metadata or data level of a table, i.e.* $\sum_{i=0}^{n} M_{xi}$ *or* $\sum_{i=0}^{n} D_{xi}$ *respectively, where* x *and* i *denote the table's level and cell position, respectively. For example, we can calculate such summation of embedding vectors corresponding to all attributes marked as "Metadata Lev. 1 HMD" in Figure 1(a).*

A crucial component of our methodology is the calculation of centroids and the aggregated vectors [48], [49] for data and metadata, which serve as essential reference points for understanding the typical angular distances between metadata and data. The centroids are calculated for three different settings: Metadata (C_{MDE}), Data-Data (C_{DE}), and Metadata-Data (C_{MDE-DE}). To calculate centroids in unsupervised manner, we used a subset of our datasets that has markup for metadata in the HTML tags. The tags are not 100% accurate and also are absent for the majority of tables (especially for VMD and deeper HMD levels). Despite this, at scale this bootstrapping method allows to create a good approximation of such aggregated and centroid vectors that if used in our contrastive learning method can yield impressive results. The script labels HMD using tags like <thead>, <tr>, <th>, and labels data using <tbody>, <tr>, <td>. For VMD labeling, it checks for bold tags/attributes or empty space characters in the first column of <td>tags. We used the corresponding labeled data to bootstrap the contrasting learning approach during the training phase by pre-calculating the metadata and data centroids. In some datasets such partial HTML tag markup may not be available (e.g., in SAUS [23] and CIUS [23]). In that case, we used the first row/column instead to calculate the metadata centroids. In the classification phase, we use these centroids and pass the entire table row-wise or column-wise to distinguish different levels of metadata *without using any labeled data*, thus ensuring no human supervision is needed.

DEFINITION 9 (METADATA TERM). *An individual term from metadata -* MD_i $\forall i = 1..n$*, where* i *is the index of the term (e.g., "Indiana" in Figure 1(a) is the metadata term).*

DEFINITION 10 (DATA TERM). *An individual term from a part of the table having data (i.e. not metadata) -* D_i $\forall i = 1..n$*, where* i *is the index of the term. For example, a cell value "19,639" in row 2, column 4 in Figure 1(a) is the data term.*

DEFINITION 11 (METADATA CENTROID). *The Metadata*

Centroid (C_{MDE}), corresponds to the range of angles across all metadata term embeddings corresponding to MD_i for all i in each metadata level. At each metadata level of the table, the embedding vectors of Metadata terms are aggregated. The minimum and maximum angles among all such vectors are then computed to calculate C_{MDE}, as shown in equation 2. This range represents the angular difference between embeddings corresponding to the metadata rows, helping to distinguish metadata rows in a new, previously unseen table.

$$C_{MDE} = [min(angle_{(m_i,m_j)}), max(angle_{(m_i,m_j)})] \quad (2)$$

where m_i and m_j are two aggregated metadata level vectors, $i, j \in (1, M_n)$, M_n is the maximum metadata level in a table and angle(m_i, m_j) represents the angle between the metadata aggregate level vectors m_i and m_j.

DEFINITION 12 (DATA-DATA CENTROID). The *Data-Data Centroid (C_{DE}) represents the range of angular distances among the aggregated data term embeddings. It is computed by aggregating the embedding vectors of data terms at each level, and then determining the minimum and maximum angles between these aggregated vectors. This range represents the observed angular difference between data rows among all tables.*

$$C_{DE} = [min(angle_{(d_i,d_j)}), max(angle_{(d_i,d_j)})] \quad (3)$$

where, d_i and d_j are two aggregated data level vectors, $i, j \in (1, D_n)$, D_n is the maximum data level in a table and angle(d_i, d_j) represents the angle between the aggregate vectors of d_i and d_j of a table.

DEFINITION 13 (METADATA-DATA CENTROID). *The Metadata-Data Centroid (C_{MDE-DE}) represents the range of angles observed between all the metadata term embeddings and data term embeddings among all tables. First, at every metadata and data level, the embedding vectors of metadata and data terms are aggregated respectively. Next, the minimum and maximum angles are determined to calculate C_{MDE-DE}.*

$$C_{MDE-DE} = [min(angle_{(m_i,d_j)}), max(angle_{(m_i,d_j)})] \quad (4)$$

where, m_i and d_j are two metadata and data level vectors, $i \in (1, M_n)$, M_n is the maximum metadata level in a table, $j \in (1, D_n)$, D_n is the maximum data level in a table and $angle(m_i, d_j)$ represents the angle between the aggregated metadata and data level vectors. This range provides a reference point for distinguishing metadata from data in previously unseen tables.

By calculating these ranges, we are creating the reference points for typical angular distances between the aggregated metadata and data embedding vectors. These references are valuable for accurate identification and classification of metadata in unsupervised manner. The centroids are calculated during the training phase and subsequently used during the classification phase. To calculate the cosine similarity between two embedding vectors $\vec{a}$ and $\vec{b}$ we use the formula [50]:

$$cos(\Theta) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|\|\vec{b}\|} \quad (5)$$

where $\vec{a} \cdot \vec{b} = \sum_{i=1}^{n} a_i b_i$ is the dot product of the two vectors, $\|\vec{a}\| = \sqrt{\sum_{i=1}^{n} a_i^2}$ is the magnitude (Euclidean norm) of vector $\vec{a}$, and $\|\vec{b}\| = \sqrt{\sum_{i=1}^{n} b_i^2}$ is the magnitude (Euclidean norm) of vector $\vec{b}$.

C. Angle Calculation

Calculation of angular distances is the crux of our methodology. We create embedding vectors for each term in our dataset. As an inherent property of embedding vectors, the angle between them corresponds to the degree of similarity or difference. Angles are used as key measures to differentiate parts of a table, specifically multi-level metadata and data. The angles are calculated for three distinct scenarios: Metadata-Metadata (Δ_{MDE}), Data-Data (Δ_{DE}), and Metadata-Data (Δ_{MDE-DE}).

DEFINITION 14 (Δ_{MDE}). *The Metadata-Metadata Angle, denoted as Δ_{MDE}, represents an angle calculated between the aggregated level vectors of two metadata rows. For each table, we calculate the summation of term embeddings at each metadata level to form the aggregate vectors. The angle between such vectors at different metadata levels reflects their semantic similarity.*

$$\Delta_{MDE} = \cos^{-1}\left(\frac{m_i \cdot m_j}{\|m_i\|\|m_j\|}\right) \quad (6)$$

where m_i and m_j are two aggregated metadata level vectors.

DEFINITION 15 (Δ_{DE}). *The Data-Data Angle, Δ_{DE}, represents the angle calculated between the aggregated level vectors of two data rows. We compute term embeddings for data terms for each table and their summation for each data row to form aggregated vectors. We then calculate the angles between such vectors at different data levels. This angle between different data levels reflects their semantic similarity.*

$$\Delta_{DE} = \cos^{-1}\left(\frac{d_i \cdot d_j}{\|d_i\|\|d_j\|}\right) \quad (7)$$

where d_i and d_j are two aggregated data level vectors.

DEFINITION 16 (Δ_{MDE-DE}). *The Metadata-Data Angle, denoted as Δ_{MDE-DE}, is an angle between metadata and data aggregated level vectors. For each table, we compute term embeddings separately for metadata and data. These embeddings are summed at their respective levels, producing metadata and data aggregated level vectors. Then we compute the angle between these vectors, which signifies the semantic difference between metadata and data in a table, key in distinguishing metadata from data.*

$$\Delta_{MDE-DE} = \cos^{-1}\left(\frac{m_i \cdot d_j}{\|m_i\|\|d_j\|}\right) \quad (8)$$

where m_i and d_j are the aggregated metadata and data level vectors.

The calculations of these angles and centroids in the previous section equip us with valuable metrics to differentiate

between metadata and data in complex generally structured tables. The angle (or cosine similarity, equation 5) between centroids and vectors measures the degree of semantic similarities between data terms in the embedding space. Angular distance as well as cosine similarity is a stable, well-established metric broadly used in research community, especially in embedding-based solutions for clustering and classification [51]–[55]. The advantage of using cosine similarity is that, it is not sensitive to the size of rows/columns in a table unlike other methods [54]. If we use only the magnitude of vectors without the angular distance, then even two rows/columns with very similar content can still exhibit a significant difference in their vectors magnitude. Other similarity metrics, such as Euclidean distance [56], measure the absolute distance between two vectors in a vector space and are sensitive to the vector's magnitude, i.e. two vectors that should be close (semantically), but have different magnitude might appear far apart. Jaccard similarity [57] is designed for set similarity and measures the direct overlap of elements between two sets, which is different from vector semantic similarity. Summation of embedding vectors is a common practice in embedding-based models [48], [49] due to its effectiveness in aggregating/combining different several components to capture the complete semantics in one vector. Concatenation of several vectors capturing different aspects of the data into a single vector is another common practice [58], [59]. It increases the resulting vector dimensionality, and preserves all individual features without any loss of information. On the other hand, summation maintains the original dimensionality, but blends the information from different vectors. It is more computationally efficient and memory friendly as well as resulted in excellent performance of our models, so we chose it over more computationally intensive concatenation [58], [59].

D. Contrastive Learning-based Metadata Identification

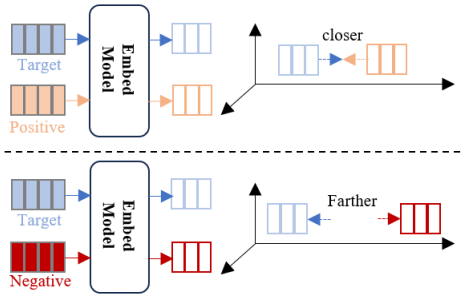

Fig. 4: Contrastive Learning.

This section delves into our process for categorizing each row and column of the table as metadata or data, leveraging the angle and centroid concepts defined in the previous sections. Figure 4 illustrates schematically the contrastive learning approach [37]. During classification, each evaluated pair consists of a "target" row/column and either a "positive" or a "negative" row/column. A (Target, Positive) pair exhibits closer aggregated row/column embeddings (e.g., two header rows or data rows), while a (Target, Negative) pair has farther aggregated row/column embeddings (e.g., a header row paired with a data row, or vice versa). The angle between positive pairs is minimized (indicating maximum similarity), whereas the angle between negative pairs is maximized (indicating minimum similarity). We use the dot product as the similarity function, following previous work [12], [60].

1) Identification of Horizontal Metadata (HMD): For identifying HMD, we consider each row of the table in a sequential manner. We start by calculating the angle between the first row and the aggregated vectors for data and metadata pre-calculated together with the centroids during the bootstrapping process. It is labeled with the label corresponding to the closest aggregated vector. Next the angle between the first and the second row is measured and if the resultant angle falls within the centroid range corresponding to metadata (C_{MDE}), the second row is classified as metadata, otherwise if the angle exceeds the metadata centroid range and falls into the metadata-data centroid range (C_{MDE-DE}), the ensuing row is classified as data. This specific shift in the angle range delineates the transition from metadata to data, simultaneously allowing us to ascertain the depth of the HMD. We then proceed to the remaining rows, maintaining the process of calculating the angle between the current and the succeeding row under consideration and classifying in the same manner. To illustrate this process, consider Figure 1 (a), where both HMD and VMD are explicitly labeled. We begin with the first row (row_1) and compute its angle with the reference metadata row (row_{mref}) and the reference data row (row_{dref}), both marked during bootstrapping. $\Delta_{row_1-row_{mref}} \in C_{MDE}$, while $\Delta_{row_1-row_{dref}} \notin C_{MDE-DE}$ or C_{DE}. Consequently, row_1 is marked as metadata and the metadata depth is updated to one. $\Delta_{row_1-row_2} \in C_{MDE-DE}$ classifying second row as data, thus keeping the metadata depth at one. For the remaining rows, $\Delta_{row_i-row_j} \in C_{DE}$, classifying them as data, where $i \in \{2, \ldots, j-1\}$ and j is the index of the last row.

2) Identification of Vertical Metadata (VMD): The identification of VMD follows a similar process to that of HMD. However, in this case, the analysis is transposed to consider columns rather than rows. We calculate the angle between each column and the subsequent column, verifying whether the angle falls within the appropriate centroid range. Depending upon whether the angle aligns with the metadata centroid range or the metadata-data centroid range, we proceed to label the column as metadata or data, respectively. To illustrate this process, consider Figure 1 (a), where both HMD and VMD are explicitly labeled. We begin with the first column (col_1) and compute its angle with the reference metadata column (col_{mref}) and the reference data column (col_{dref}), both marked during bootstrapping. $\Delta_{col_1-col_{mref}} \in C_{MDE}$, while $\Delta_{col_1-col_{dref}} \notin C_{MDE-DE}$ or C_{DE}. Consequently, col_1 is marked as metadata and the metadata depth is updated to one. $\Delta_{col_1-col_2} \in C_{MDE}$ and $\Delta_{col_2-col_3} \in C_{MDE}$ classify the second and third columns as metadata, updating the metadata depth to three. $\Delta_{col_3-col_4} \in C_{MDE-DE}$ classifying col_4 as data. For the remaining columns, $\Delta_{col_i-col_j} \in C_{DE}$,

Algorithm 1: Metadata classification in Generally Structured Tables (GST).

Input: Generally Structured Table T
Output: Labeled rows and metadata depth

Convert terms in T to embeddings E using Word2Vec or BioBERT
Calculate Centroid ranges for metadata (C_{MDE}), metadata-data (C_{MDE-DE}) and data-data (C_{DE})
for *each row* r_i *in* T **do**
    for *each row* r_j *in* T *where* $j > i$ **do**
        Compute angle Δ_{ij} between rows r_i and r_j
        if Δ_{ij} *in* C_{MDE} **then**
            Label r_j as metadata
        end
        else if Δ_{ij} *in* C_{MDE-DE} **then**
            Label r_j as data
            Break
        end
    end
end
Repeat the process for columns to identify vertical metadata
return *Labeled rows and depth of metadata*

classifying them as data, where $i \in \{4, \ldots, j-1\}$ and j is the index of the last column.

Above, we discussed our unsupervised algorithms to identify both horizontal and vertical metadata in complex generally structured tables at scale. Algorithm 1 encodes the steps, starting from creating embeddings, to calculating Centroid ranges, angles, and labeling rows and columns based on their calculated angular distances to Centroids. By iterating over each row and, subsequently, over each column, the algorithm can not only effectively label, but also determine the depth of metadata in unsupervised manner.

IV. Experimental Evaluation

In this section, we describe the experimental evaluation of our unsupervised contrastive learning approach on 6 large-scale datasets.

A. Infrastructure

We ran our experiments on a cluster of servers with Intel Xeon CPUs, from 192GB to 1TB of RAM, 10TB disk space each. BioBERT [47] was fine-tuned using PyTorch framework; Word2Vec [46] - using Gensim.

B. Datasets

To compare to SOTA, we used 6 large-scale datasets.

- CORD-19 [2]: The CORD-19 dataset is a collection of papers related to SARS and MERS viruses [2]. Tables used in the papers are extracted from PDF and stored in JSON format. CORD-19 has $\approx 130K$ medical generally structured tables abundant in HMD and VMD, both regular and hierarchical.
- CKG: CKG is extracted from medical publications (up to 2022) on COVID-19, obtained via PubMed.com. It contains 0.82 million papers and 1.4 million tables. The tables exhibit both VMD and HMD.
- CIUS [61]: The CIUS dataset is from the Crime In the US (CIUS) database and consists of 1,050 tables, including information about crime offenses.
- SAUS [61]: The 2010 Statistical Abstract of the United States (SAUS) comprises 1,368 tables which can be downloaded from the U.S. Census Bureau. It covers a variety of topics, including finance, business, crime, agriculture, and health care.
- Web Data Commons (WDC) [1]: This dataset consists of more than 100M Web tables (more than 15M in English) from 265K sources, including information about their source URL, table type, and text before and after the table. The tables are from a variety of domains, including scientific, news articles, product information, and many others. We took a subset of 100K tables for our experiments.
- PubTables-1M [32]: This dataset contains nearly 1 million tables from scientific articles in the PubMed Central Open Access database.

C. Training Embedding Models

To create term embeddings, we took the vocabulary and pre-trained embedding weights from BioBERT and fine-tuned it on our training datasets. The configuration of BioBERT is aligned with the $\text{BERT}_{\text{BASE}}$ model [44], which uses a 768-dimensional space for embedding. We ran training for 50,000 steps, with a batch size of 12, and a learning rate of 2×10^{-5} for 3 days. The training set is comprised of table tuples/rows. We tokenize, embed, encode each tuple and use fixed maximum sequence length of 512 as described in [47]. We add *[CLS]* at the start of each row and *[SEP]* between the cells [47]. We also trained Word2Vec embedding model [46] on our datasets with embedding dimensionality 300, the context window of size 3 before and after the target word, minimum count of 1 for word inclusion. We did experiments with several embedding dimensions and found no notable performance difference when using the embeddings trained with the dimension more than 300. However, the slowdown in training time was significant so we chose 300 as optimal dimensionality.

D. Baseline Methods

We compare our method to 3 well-known SOTA approaches for table header detection and classification.

- Pytheas [21] is a method for classifying lines in a CSV file into header, data and subheader, using fuzzy logic (see Related Work). Their approach achieves 95.13% precision for HMD level 1 (on their datasets) and 88.14% "subheader" precision (CMD according to our definitions, not the deeper levels of HMD), but does not support

TABLE I: Centroid and Angles for Identifying Levels 2-5 of Horizontal Metadata. MDL - Meta Data Level.

Dataset	MDL	Centroid$_{MDE,DE}$	Centroid$_{DE,DE}$	Centroid$_{MDE,MDE}$	$\Delta_{1MDE,2MDE}$	$\Delta_{2MDE,DE}$
CKG	Lev. 2	65 to 75	25 to 35	15 to 45	37	62
CORD-19	Lev. 2	60 to 75	20 to 35	25 to 45	29	58
CIUS	Lev. 2	55 to 65	24 to 35	25 to 40	32	60
SAUS	Lev. 2	56 to 66	26 to 35	20 to 40	38	65
Dataset	**MDL**	**Centroid**$_{MDE,DE}$	**Centroid**$_{DE,DE}$	**Centroid**$_{MDE,MDE}$	$\Delta_{2MDE,3MDE}$	$\Delta_{3MDE,DE}$
CKG	Lev. 3	50 to 65	25 to 35	25 to 45	40	61
CORD-19	Lev. 3	51 to 60	20 to 35	30 to 42	33	58
SAUS	Lev. 3	50 to 58	26 to 35	20 to 35	35	55
Dataset	**MDL**	**Centroid**$_{MDE,DE}$	**Centroid**$_{DE,DE}$	**Centroid**$_{MDE,MDE}$	$\Delta_{3MDE,4MDE}$	$\Delta_{4MDE,DE}$
CKG	Lev. 4	45 to 50	25 to 35	20 to 40	40	65
CORD-19	Lev. 4	46 to 51	20 to 35	30 to 42	40	46
Dataset	**MDL**	**Centroid**$_{MDE,DE}$	**Centroid**$_{DE,DE}$	**Centroid**$_{MDE,MDE}$	$\Delta_{4MDE,5MDE}$	$\Delta_{5MDE,DE}$
CKG	Lev. 5	55 to 65	25 to 35	20 to 30	40	65

TABLE II: Centroid and Angles for Identifying Level 1 HMD.

Dataset	Centroid$_{MDE,DE}$	Centroid$_{DE,DE}$	$\Delta_{MDE,DE}$
CORD-19	60 to 75	25 to 35	65
CKG	65 to 75	25 to 35	65
WDC	65 to 75	20 to 35	60
CIUS	75 to 98	24 to 35	90
SAUS	60 to 70	26 to 35	60
PubTables	70 to 85	35 to 40	75

TABLE III: Centroid and Angles for Identifying Level 1 VMD.

Dataset	Centroid$_{MDE,DE}$	Centroid$_{DE,DE}$	$\Delta_{MDE,DE}$
CORD-19	45 to 62	27 to 35	52
CKG	45 to 60	25 to 35	55
WDC	45 to 65	25 to 35	50
CIUS	40 to 60	24 to 35	55
SAUS	40 to 60	26 to 35	50

VMD classification (Table V) and does not distinguish between HMD levels, so we cannot fully compare to it.

- [22] adopts a Random Forest classifier to detect and classify table headers. However, the authors do not provide the publicly available code, hence we were unable to evaluate their method on our datasets and include the results along with other methods in Table V. Tables in their datasets contain VMD up to 2 levels and HMD up to 3 levels. Their reported accuracy on their datasets - 92% for HMD (monolithically, without identifying any separate levels), 90.4% for VMD (again monolithically).
- Table Transformer (TT) [32] is a deep learning model based on object detection that recognizes tables from images. One of its subtasks is Table Structure Recognition (TSR), which identifies a table's structure using six object classes: table, table column, table row, table column header, table projected row header, and table spanning cell. It does not distinguish between HMD levels and does not support VMD classification (Table V), so we cannot fully compare it to our results.

E. Evaluation Metric

We evaluated the performance of our approach for metadata classification using the accuracy metric [62], [63].

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{9}$$

where TP, TN, FP, and FN denote the true positives, true negatives, false positives, and false negatives, respectively, as explained in more detail in [62].

F. Results and Analysis

The experimental results are summarized in Table V. Our approach outperforms SOTA on all large-scale datasets on location and classification of VMD at any level (3 was the deepest hierarchy level found in all datasets). The centroid and angles calculated for each dataset are summarized in Tables I–IV.

Identifying Horizontal Metadata (HMD): Figure 6 comparatively illustrates performance our method for HMD classification across 6 datasets for 5 levels of HMD. We noted that deeper-level metadata beyond 1-2 is less common, but is abundant in our datasets.

Given the ubiquity of level 1 HMD, we conducted experiments for its detection across all six datasets. Our approach involved computing metrics, such as Centroid$_{\text{MDE, DE}}$, Centroid$_{\text{DE, DE}}$, and Centroid$_{\text{MDE, MDE}}$, along with $\Delta_{\text{1MDE, DE}}$. As previously defined, centroids provide insights into the average relative angular position of metadata versus data. Additionally, the calculated angles between different metadata levels are used to infer the hierarchical relationship between metadata levels as well as the boundary between data and metadata. The centroids and angles are calculated separately for each dataset.

For the identification of level 2 HMD, we employed similar centroid measurements as in level 1 HMD experiments, but adjusted the angle calculations to reflect the different level of hierarchy, including $\Delta_{\text{1MDE, 2MDE}}$ and $\Delta_{\text{2MDE, DE}}$. Notably, the WDC dataset was excluded from this experiment due the sparsity of high quality tables in English with level 2 and deeper-level HMD in it.

In the case of level 3 HMD identification, we computed angles between level 2 and level 3 HMD, and between level 3 HMD and Data—$\Delta_{\text{2MDE, 3MDE}}$ and $\Delta_{\text{3MDE, DE}}$. Figure 5 illustrates an example for a table in CKG dataset.

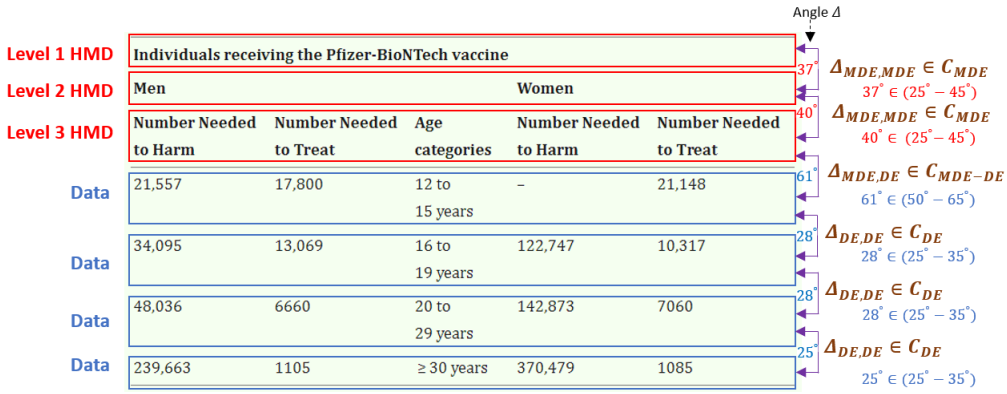

Fig. 5: A Sample Table with the Classified HMD (Levels 1-3), Centroids, and Deltas.

For level 4 HMD, the experiments utilized two datasets, CORD-19 and CKG, with each dataset having $\approx 1K$ tables with HMD level 4. We calculated the angles between level 3 and level 4 HMD, as well as between level 4 HMD and Data—$\Delta_{3MDE,\ 4MDE}$ and $\Delta_{4MDE,\ DE}$.

Level 5 HMD is relatively rare, but still exists, especially in medical research literature. Our experiment was conducted on CKG dataset, with ≈100 tables having HMD level 5. The angle measurements taken were between level 4 and level 5 HMD, and between level 5 HMD and Data—$\Delta_{4MDE,\ 5MDE}$ and $\Delta_{5MDE,\ DE}$.

Identifying Vertical Metadata (VMD): We conducted a series of experiments focused on identifying VMD at different columnar depth. The deepest hierarchy level we found in VMD in all our datasets was three. Figure 7 comparatively illustrates performance of our methods for VMD identification up to level 3, across 5 datasets. For the identification of VMD we employed similar centroid calculations as in HMD described above i.e. $\text{Centroid}_{MDE,\ DE}$, $\text{Centroid}_{DE,\ DE}$, and $\text{Centroid}_{MDE,\ MDE}$. The first experiment was to identify level 1 VMD. Given the absence of a preceding metadata level in this case, the angle between metadata levels is not applicable. We computed the angle between level 1 VMD (first column) and Data column(second column)—$\Delta_{1MDE,\ DE}$. The centroid and angle calculated for this experiment are summarized in Table III. Next, we aimed to identify level 2 VMD. This involved assessing the relationship of level 2 metadata with the level 1 metadata and the data—$\Delta_{1MDE,\ 2MDE}$ and $\Delta_{2MDE,\ DE}$. Finally for the identification of level 3 VMD, the angle calculated was between level 2, level 3 metadata, and data—$\Delta_{2MDE,\ 3MDE}$ and $\Delta_{3MDE,\ DE}$. The centroid and angle calculated for VMD levels 2-3 are summarized in Table IV.

Analysis: From the experimental results in Table V, we observe that our method effectively identifies complex multi-level horizontal and vertical metadata across six datasets of varying complexity, demonstrating its adaptability to heterogeneous data sources as well as scalability. Our approach achieves high-accuracy on VMD classification, reaching up to 96%, compared to the SOTA maximum of 90.4% reported by [22], but only for two first levels of VMD combined. Our method

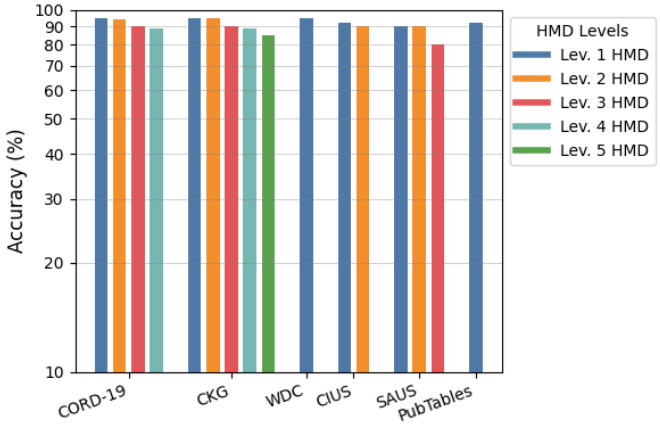

Fig. 6: Accuracy of HMD Detection, Levels 1-5.

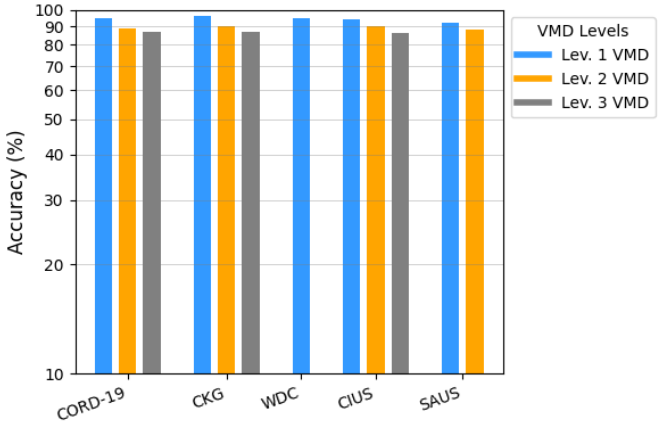

Fig. 7: Accuracy of VMD Identification, Levels 1-3.

significantly outperforms SOTA on the remaining, deeper VMD levels, whereas Pytheas [21] and Table Transformer [32] do not support VMD classification at all.

For HMD level 1, Pytheas achieves a slightly higher accuracy than our approach, with an accuracy delta of approximately 3%. However, we surpass all baselines for all

TABLE IV: Centroid and Angle Calculations for Identifying Levels 2-3 of Vertical Metadata. MDL - Meta Data Level.

Dataset	MDL	Centroid$_{MDE,DE}$	Centroid$_{DE,DE}$	Centroid$_{MDE,MDE}$	$\Delta_{1MDE,2MDE}$	$\Delta_{2MDE,DE}$
CORD-19	Lev.2	45 to 62	25 to 35	35 to 45	38	45
CKG	Lev. 2	45 to 60	20 to 35	30 to 42	40	46
CIUS	Lev. 2	40 to 60	24 to 35	25 to 40	38	45
SAUS	Lev. 2	40 to 60	26 to 35	25 to 40	35	40
Dataset	**MDL**	**Centroid**$_{MDE,DE}$	**Centroid**$_{DE,DE}$	**Centroid**$_{MDE,MDE}$	$\Delta_{2MDE,3MDE}$	$\Delta_{3MDE,DE}$
CORD-19	Lev. 3	45 to 62	26 to 36	24 to 32	37	52
CKG	Lev. 3	45 to 60	25 to 35	20 to 30	35	50
CIUS	Lev. 3	40 to 60	24 to 35	20 to 30	35	50

TABLE V: Accuracy in % for Identifying Levels 1-5 of HMD/Levels 1-3 of VMD. In Meta Data Level the subscript indicates the level/depth. A '-' indicates that the method does not distinguish between different levels of Metadata and does not support VMD classification. TT - Table Transformer.

Dataset	Meta Data Level	Pytheas	TT	Our method
CORD-19	HMD_1/VMD_1	98/-	88.3/-	95/95
	HMD_2/VMD_2			94/89
	HMD_3/VMD_3			90/86.8
	HMD_4			89
CKG	HMD_1/VMD_1	98/-	84.2/-	95/96
	HMD_2/VMD_2			95/90
	HMD_3/VMD_3			90/87
	HMD_4			89
	HMD_5			85
WDC	HMD_1/VMD_1	96/-	83.1/-	94.5/95
CIUS	HMD_1/VMD_1	98/-	87.5/-	92/94
	HMD_2/VMD_2			90/90
	VMD_3			86
SAUS	HMD_1/VMD_1	96/-	87.6/-	90/92
	HMD_2/VMD_2			90/88
	HMD_3			80
PubTables	HMD	98/-	91.2/-	92

remaining HMD levels 2-5. More importantly, the baselines do not distinguish between different HMD levels at all as well as require supervision. These results indicate that our proposed contrastive learning approach is particularly effective in identifying deep levels of hierarchical metadata, where SOTA supervised methods tend to fall short. SOTA solutions do not explicitly separate VMD and HMD or classify multi-layer hierarchical metadata in complex non-relational tables, making it difficult to query such tables correctly (i.e. according to their schema) and limiting the user's ability to *understand* the structural intricacies of such tables. Accurate identification of both HMD and VMD is essential for fine-grained structural query processing, correct data access, and efficient structural search. Without this, valuable structural information is lost, impacting query processing and all downstream tasks relying on table schema, such as question answering on tables, data analysis, information extraction, data fusion, and many other. Our method addresses these challenges by enabling precise hierarchical metadata identification, thus avoids hindering performance as noted above.

G. RunTime

We measured the training and inference times on a 40-core Intel Xeon E7-4870 2.40 GHz CPU with 192GB of RAM.

Training: The training on 200K tables (see Section IV-B) took $\approx$ 3 days. To compare, Pytheas took $\approx$ 1.5 days, while Table Transformer took $\approx$ 2 days on the same hardware. Notably, these baselines rely on manual annotation, which takes additional time and incurs cost, hence total duration will be significantly longer as well as will have an additional expense for annotation compared to our approach.

Inference: The inference runtime for both our method and the baseline methods scales approximately linearly with the dataset size. Pytheas processes files at 0.021 sec per column, per row, i.e. for a table with 5 columns and 10 rows, it takes 1.05 sec. Table Transformer averages 1.56 sec per table, whereas our approach takes on average 1.8 sec per such table. Although our method has additional computational overhead due to embedding-based processing, both baselines and our approach scale linearly as the dataset size increases.

Hybrid solution: To further improve efficiency, one can first apply SOTA techniques to identify metadata in simpler relational tables (i.e., those with a single level of HMD), and then, for the remaining tables employ our approach, where accurate classification of Bi-dimensional hierarchical metadata (i.e, HMD, VMD with multiple levels exhibiting hierarchical relationships, as shown in Figure 1) justifies the additional expense.

H. LLMs for HMD and VMD classification

Dataset: Due to the very high cost (especially GPT-4) to experiment with all datasets, we had to pick a good representative example. CKG dataset contains various types of tables with different levels of HMD/VMD. We selected a random sample from the CKG, each representing different levels/depths (see Def 7) of HMD and VMD.

Pre-processing: The tables were pre-processed to standardize the format before being input into the GPT-based automated labeling system. Pre-processing included aligning rows and columns, and removing any corrupt or unreadable data.

Input Format to the LLM: The LLM receives table data input in a standardized CSV (Comma-Separated Values) format, typical for database and spreadsheet applications.

LLM setup: We configured the LLMs (GPT 3.5 and 4) [64], [65] as an automated system capable of understanding and processing table data, with the specific task of emulating the capabilities of a database administrator skilled in labeling tables into HMD, VMD, and Table Data. For this, we feed the system-level message via the LLM's API describing

its expected behavior as:"*You are a helpful assistant who understands table data. The general table structure is as follows: HMD generally includes the first row, but can extend to multiple rows depending on the table structure; VMD consists of the vertical headers, which may include one or more columns; any remaining rows/columns are classified as Table Data*". This explicit definition of the LLM's role with information on the general structure of tables allows it to understand generally structured table structure and perform the task submitted by the user in the next step.

Interaction with the LLM: We interact with the LLM by submitting a request to label the provided table. We specify the request through a structured prompt by detailing the total number of rows and columns (optional), followed by the data entries formatted as plain text or CSV. An example prompt would be: *"I am giving you table data. Please provide labels for HMD, VMD, and Data, i.e., what each row belongs to. Below are my rows for the table. It has 9 rows and 6 columns followed by the 'Table data' ..."*.

LLM Response to User Prompt: Upon receiving the user prompt with the table data, the LLM performs labeling and provides the labels. For a table with three columns and multiple rows, the system might output the following labels: *"HMD: 'Row 1: Column1, Column2, Column3' VMD: 'Column1, Column2' Table Data: All data entries from Row 2 onwards under each column"*.

To summarize, LLMs slightly outperform us with a delta of 4% in accuracy for HMD level 1, but we outperformed LLMs with delta of 29% for all other levels 2-5 HMD and up to 87% delta for VMD (see Table VI).

We also gained the following insights from our experiments. LLM has difficulties correctly classifying HMD and VMD containing numbers (decimals, floating numbers, or percentages) and misclassify them as Table Data rather than metadata. However, the LLM's response is not consistent in this case. It can recognize these as HMD if the numbers are enclosed in parentheses or associated with keywords like 'total', 'number of', or 'percentage'. LLM struggles with accurately identifying CMD (located in the middle of the table). Additionally, LLM occasionally misidentifies ranges as HMD, and the accuracy of this detection varies. There are also instances where the same HMD label is duplicated, erroneously suggesting that the LLM has correctly identified multiple levels of metadata. In some cases, attributes from the first level of headers are incorrectly split and assigned to the second level.

I. Using LLMs+RAG for HMD and VMD Classification

We also conducted experiments using Retrieval-Augmented Generation (RAG) to enhance the quality of LLM responses. We used PubMed API to submit a RAG query [33] to PubMed.com to identify articles containing tables with a variety of formats and structures that are similar or relevant to the user's query. The results documents are in XML format. We parsed XML to get HTML code for the tables. These retrieved tables in HTML sometimes have HTML tags that tag HMD, which would help LLM to correct its mistakes. The reason for using PubMed.com is that the tables in our CKG dataset, which we input into the LLM, are sourced from the articles published on PubMed.com. When a user submits a table through the structured prompt that we have implemented, the RAG system fetches such table (if it exists) from our database. This table with its HTML code is then fed into the LLM along with the initial query. We can see from the results that such RAG improves LLM's performance.

LLM+RAG outperforms LLM on identifying level 2 HMD by a significant delta of 12% on CKG (Table VI). LLM+RAG again outperforms LLM with a large delta 15% on CKG to identify level 3 VMD where, accuracy was 0% without RAG. LLMs+RAG slightly outperform us on HMD level 1 with a delta in accuracy of 5%, while we outperform it with deltas up to 12% for HMD levels 2-5 and up to 15% delta for VMD.

TABLE VI: Accuracy in % for identifying HMD/VMD on CKG dataset. In Metadata Level subscript indicates the level.

Metadata Level	GPT3.5	GPT4	RAG+GPT4
HMD_1/VMD_1	98/52	99/70	100/81
HMD_2/VMD_2	60/16	70/50	82/56
HMD_3/VMD_3	60/0	66/0	75/15
HMD_4	60	60	70
HMD_5	60	60	68

V. Related work

Accurate, scalable, generalizable Metadata location and classification in Web Tables, CSV files, tables extracted from scientific publications, and other large-scale structured corpora is gaining momentum in the Data Management and Science communities [18], [21], [22], [66], [67]. It is due to both the fundamental nature of Metadata and its being of critical importance for many downstream data management tasks. Several studies have primarily focused on developing efficient techniques for categorizing metadata based on multiple criteria and features. Some of the recent work utilizing the supervised and unsupervised learning approach is discussed below.

Pytheas [21] proposed a supervised line classification system for CSV files that uses fuzzy logic to determine whether a field is data or not. It operates in two phases: an offline (training) phase, where it learns rule weights, and an online (inference) phase, where it assigns confidence scores to lines. Evaluated on two manually annotated datasets, Pytheas achieved 95.6% accuracy in table row classification (for both data and HMD rows *combined*, so it cannot be directly compare with our HMD classification accuracy); 95.13% precision for HMD level 1, and 88.14% precision for "subheader" (CMD according to our definitions, not the deeper levels of HMD). However, it does not support VMD classification or deeper HMD levels (2-5). While Pytheas slightly outperforms us on HMD level 1 (by 0.18%), its evaluation on only two sources limits claims about its scalability to datasets composed from many sources [25], [68]. The authors in [22] use Random Forest to detect and classify table headers, proposing two

heuristic strategies to separate data from the header. They use the first row and column as baseline headers, achieving 92% accuracy for HMD level 1-3 *combined* and 90.4% for VMD level 1-2 *combined*. We outperform them on VMD, match or exceed their performance on HMD level 1-3 across all datasets except SAUS, and outperform them on HMD level 4-5. Several studies have explored supervised algorithms for tabular data classification. TabPFN [69] is a Transformer-based model for fast classification of small datasets. GATE [70] is a parameter-efficient deep learning architecture using gating and decision tree ensembles for feature selection. [71] proposed a hybrid probabilistic approach for table understanding, dividing it into cell function classification, block detection, and layout prediction using table cell embeddings from [72]. It analyzes table structure by identifying relationships between spreadsheet blocks (regions with similar cell types). However, since the authors did not release their code, we could not include it as a baseline for comparison. Other works [73]–[83] use supervised learning to capture complex relationships and identify key features. However, these methods rely on large labeled datasets and struggle with overfitting and lack semantic/contextual understanding, particularly in metadata classification. These limitations motivate the need for approaches that capture contextual, semantic, and positional features to classify metadata in heterogeneous tabular data.

GReat [38] leverages an auto-regressive language model to generate synthetic tabular data, incorporating textual encoding but doesn't address metadata classification or scalability for complex datasets. Similarly, TabDDPM [39], a diffusion-based model, excels in data augmentation and missing value imputation but overlooks metadata classification. VIME [40] employs self- and semi-supervised learning for unlabeled tabular data, yet struggles with scalability. TabNet [41] uses attention for sequential feature selection but faces computational challenges and fails to address metadata classification. SuperTM [42] applies two-dimensional word embeddings to tabular data but struggles with noisy data and scalability for large datasets. While these methods advance unsupervised and self-supervised learning for tabular data, they underscore the need for an unsupervised model that improve performance and address metadata classification.

The table discovery system serves data lake tables to end users (e.g., data scientists) by enabling them to query and explore tables relevant to their downstream analytic tasks. [84] introduce an attribute-unionability framework that assesses table similarity by evaluating attribute relatedness. Aurum [85] leverages an enterprise knowledge graph (EKG) to capture and query relationships among datasets in data lakes, focusing on indexing and keyword search to identify related datasets based on simple term matching from user queries. Alternatively, some discovery systems support data navigation, allowing users to explore tables in a data lake using an automatically discovered organization of the tables. For instance, [86] employ a lineage graph to manage tables in a data lake. All of these approaches rely on table metadata within the data lake to identify relevant tables or tuples based on search keywords. However, in real-world large-scale data lakes, metadata is not perfect—it may be incomplete, inconsistent, or entirely missing. In this work, we focus on improving and scaling up accurate metadata identification and classification in tabular data to help support accurate structural search. Structural search in data lakes could make table search and discovery more precise and accurate compared to just keyword-search (used in absence of high-quality metadata) that usually blindly treats the all table sections as data.

[18] recently performed line and cell classification in verbose CSVs, focusing on CSV structure detection by identifying cell types such as metadata, header, group, data, derived, notes etc. Our work is similar to the part of their work on cell classification, more precisely, the header cell classification. For cell classification, they have used content, contextual and computational features of the cell. Specifically, they have analyzed the number of empty cells, the position of the row or column, whether there is any empty column besides the column being analyzed, block size, data type, etc. This analysis is, however, purely cell-based and does not take into account the compositional features of cells when they become tuples or columns as well as the context, which we do. Another line of work, Starmie [87] transforms columns into sequences and fine-tunes BERT [44] using a contrastive learning framework, where unionable and non-unionable column pairs serve as training samples to find all tables that are unionable with the query table. In contrast, our work focuses on metadata classification in tables. In [34], two scalable binary metadata classification architectures were introduced: one employing an SVM model with a polynomial kernel that incorporates novel positional features for each row and column of the table, and another using an ensemble of BiGRU models with parallel layers of embeddings. Both approaches are supervised. In contrast, the approach described in this paper is unsupervised and does not require labor-intensive labeling.

VI. Conclusion

In this paper, we address an important and challenging problem of identifying and classifying complex multi-layer, hierarchical vertical and horizontal metadata in Generally Structured Tables (GST). We present an unsupervised, scalable contrastive-learning approach and evaluate it on 6 large-scale, heterogeneous datasets composed from thousands to millions of sources. We outperform SOTA and LLMs in classifying horizontal metadata (HMD) of deep levels (3-5) and for all levels (1-3) of vertical metadata (VMD). LLMs with/without RAG slightly outperform us with deltas of 4-5% in accuracy for HMD level 1, but we significantly outperformed LLMs/LLMs+RAG with delta up to 29% for all other levels 2-5 HMD and up to 87% delta for VMD.

References

[1] O. Lehmberg, D. Ritze, R. Meusel, and C. Bizer, "A large public corpus of web tables containing time and context metadata," in *WWW*, J. Bourdeau, J. Hendler, R. Nkambou, I. Horrocks, and B. Y. Zhao, Eds., 2016.

[2] L. L. Wang, K. Lo, Y. Chandrasekhar, R. Reas, J. Yang, D. Burdick, D. Eide, K. Funk, Y. Katsis, R. Kinney *et al.*, "Cord-19: The covid-19 open research dataset," *ArXiv*, 2020.

[3] "Census bureau," https://www.census.gov/data/datasets.html, 2022, accessed: 2024-05-06.

[4] M. Gubanov and A. Pyayt, "Readfast: High-relevance search-engine for big text," in *ACM CIKM*, 2013.

[5] B. Alexe, , M. A. Hernandez, H. Ho, J.-W. Huang, Y. Katsis, and L. Popa, "Simplifying information integration: Object-based flow-of-mappings framework for integration," in *BIRTE*, 2009.

[6] M. Gubanov and A. Pyayt, "Type-aware web search," in *EDBT*, 2014.

[7] M. Podkorytov, D. Soderman, and M. N. Gubanov, "Hybrid.poly: An interactive large-scale in-memory analytical polystore," in *ICDM Workshops*. IEEE Computer Society, 2017, pp. 43–50.

[8] M. Gubanov, "Hybrid: A large-scale in-memory image analytics system," in *CIDR*, 2017.

[9] S. Ortiz, C. Enbatan, M. Podkorytov, D. Soderman, and M. Gubanov, "Hybrid.json: High-velocity parallel in-memory polystore json ingest," in *IEEE Bigdata*, 2017.

[10] M. Simmons, D. Armstrong, D. Soderman, and M. Gubanov, "Hybrid.media: High velocity video ingestion in an in-memory scalable analytical polystore," in *IEEE Bigdata*, 2017.

[11] S. Soderman, A. Kola, M. Podkorytov, M. Geyer, and M. Gubanov, "Hybrid.ai: A learning search engine for large-scale structured data," in *WWW*, 2018.

[12] G. Shrestha, C. Jiang, S. Akula, V. Yannam, A. Pyayt, and M. N. Gubanov, "Tabular embeddings for tables with bi-dimensional hierarchical metadata and nesting," in *Proceedings 28th International Conference on Extending Database Technology, EDBT 2025, Barcelona, Spain, March 25-28, 2025*, 2025, pp. 92–105.

[13] M. Gubanov and M. Stonebraker, "Large-scale semantic profile extraction," in *EDBT*, 2014.

[14] M. Gubanov, A. Pyayt, and L. Shapiro, "Readfast: Browsing large documents through ufo," in *IRI*, 2011.

[15] M. Gubanov, L. Shapiro, and A. Pyayt, "Learning unified famous objects (ufo) to bootstrap information integration," in *IRI*, 2011.

[16] M. Gubanov, M. Priya, and M. Podkorytov, "Cognitivedb: An intelligent navigator for large-scale dark structured data," in *WWW*, 2017.

[17] M. Gubanov and L. Shapiro, "Using unified famous objects (ufo) to automate alzheimer's disease diagnostics," in *BIBM*, 2012.

[18] F. N. Lan Jiang, Gerardo Vitagliano, "Structure detection in verbose csv files." EDBT, March 2021.

[19] B. Hancock, H. Lee, and C. Yu, "Generating titles for web tables," in *WWW*. New York, NY, USA: Association for Computing Machinery, 2019.

[20] M. J. Cafarella, A. Halevy, D. Z. Wang, E. Wu, and Y. Zhang, "Webtables: exploring the power of tables on the web," in *VLDB*, 2008.

[21] C. Christodoulakis, E. B. Munson, M. Gabel, A. D. Brown, and R. J. Miller, "Pytheas: pattern-based table discovery in csv files," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 2075–2089, 2020.

[22] J. Fang, P. Mitra, Z. Tang, and C. L. Giles, "Table header detection and classification," *AAAI*, vol. 26, no. 1, Jul. 2012. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/8206

[23] B. Alexe, M. Gubanov, M. A. Hernández, C. T. H. Ho, J. Huang, Y. Katsis, L. Popa, B. Saha, and I. Stanoi, "Simplifying information integration: Object-based flow-of-mappings framework for integration," in *BIRTE*, 2008.

[24] M. Gubanov, C. Jermaine, Z. Gao, and S. Luo, "Hybrid: A large-scale linear-relational database management system," in *MIT NEDB*, 2016.

[25] R. Khan and M. Gubanov, "Weblens: Towards interactive large-scale structured data profiling," in *CIKM*, 2020.

[26] ——, "Weblens: Towards interactive web-scale data integration, training the models," in *IEEE Big Data*, 2020.

[27] ——, "Nested dolls: Towards unsupervised clustering of web tables," in *IEEE Big Data*, 2018.

[28] ——, "Towards unsupervised web tables clustering." in *IEEE BigData*, 2018.

[29] A. Kola, H. More, S. Soderman, and M. Gubanov, "Generating unified famous objects (ufos) from the classified object tables," in *IEEE Big Data*, 2017.

[30] S. Villasenor, T. Nguyen, A. Kola, S. Soderman, and M. Gubanov, "Scalable spam classifier for web tables," in *IEEE Big Data*, 2017.

[31] B. Alexe, M. Gubanov, M. A. Hernandez, H. Ho, J.-W. Huang, Y. Katsis, and L. Popa, "Simplifying information integration: Object-based flow-of-mappings framework for integration," in *Business Intelligence for the Real Time Enterprise*. Springer, 2009.

[32] B. Smock, R. Pesala, and R. Abraham, "PubTables-1M: Towards comprehensive table extraction from unstructured documents," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 4634–4642.

[33] "Aryn-ai/sycamore: Sycamore is an llm-powered search and analytics platform for unstructured data," https://github.com/aryn-ai/sycamore, 2024, accessed: 2024-06-06.

[34] B. Kandlibedala, A. Pyayt, C. Caballero, and M. Gubanov, "Scalable hierarchical metadata classification in heterogeneous large-scale datasets," 2023.

[35] E. F. Codd, "Further normalization of the data base relational model," *Data base systems*, vol. 6, pp. 33–64, 1972.

[36] ——, "A relational model of data for large shared data banks," *Communications of the ACM*, vol. 13, pp. 377–387, 1970.

[37] N. Reimers, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[38] V. Borisov, K. Seßler, T. Leemann, M. Pawelczyk, and G. Kasneci, "Language models are realistic tabular data generators," 2023.

[39] A. Kotelnikov, D. Baranchuk, I. Rubachev, and A. Babenko, "Tabddpm: Modelling tabular data with diffusion models," 2022.

[40] J. Yoon, Y. Zhang, J. Jordon, and M. van der Schaar, "Vime: Extending the success of self-and semi-supervised learning to tabular domain," *Advances in Neural Information Processing Systems*, vol. 33, pp. 11 033–11 043, 2020.

[41] S. Arik and T. Pfister, "Tabnet: Attentive interpretable tabular learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 8, pp. 6679–6687, May 2021. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/16826

[42] B. Sun, L. Yang, W. Zhang, M. Lin, P. Dong, C. Young, and J. Dong, "Supertml: Two-dimensional word embedding for the precognition on structured tabular data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0.

[43] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013.

[44] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[45] R. Brochier, A. Guille, and J. Velcin, "Global vectors for node representations," in *The World Wide Web Conference*. ACM, may 2019. [Online]. Available: https://doi.org/10.1145%2F3308558.3313595

[46] S. I. C. K. C. G. S. D. J. Mikolov, T., "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013.

[47] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "Biobert: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.

[48] X. Deng, H. Sun, A. Lees, Y. Wu, and C. Yu, "Turl: Table understanding through representation learning," *ACM SIGMOD Record*, vol. 51, no. 1, pp. 33–40, 2022.

[49] J. Herzig, P. K. Nowak, T. Mueller, F. Piccinno, and J. Eisenschlos, "Tapas: Weakly supervised table parsing via pre-training," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 4320–4333.

[50] M. R. Hasan and J. Ferdous, "Dominance of ai and machine learning techniques in hybrid movie recommendation system applying text-to-number conversion and cosine similarity approaches," *Journal of Computer Science and Technology Studies*, vol. 6, no. 1, pp. 94–102, 2024.

[51] H. Schütze, C. Manning, and P. Raghavan, "Introduction to information retrieval cambridge university press cambridge," 2008.

[52] F. Rahutomo, T. Kitasuka, M. Aritsugi *et al.*, "Semantic cosine similarity," in *The 7th international student conference on advanced science and technology ICAST*, vol. 4, no. 1. University of Seoul South Korea, 2012, p. 1.

[53] T. Thongtan and T. Phienthrakul, "Sentiment classification using document embeddings trained with cosine similarity," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, 2019, pp. 407–414.

[54] P. Y. Ristanti, A. P. Wibawa, and U. Pujianto, "Cosine similarity for title and abstract of economic journal classification," in *2019 5th inter-*

national conference on science in information technology (ICSITech). IEEE, 2019, pp. 123–127.

[55] B. Li and L. Han, "Distance weighted cosine similarity measure for text classification," in *Intelligent Data Engineering and Automated Learning–IDEAL 2013: 14th International Conference, IDEAL 2013, Hefei, China, October 20-23, 2013. Proceedings 14*. Springer, 2013, pp. 611–618.

[56] F. Szabo, *The linear algebra survival guide: illustrated with Mathematica*. Academic Press, 2015.

[57] H. JM, "Jaccard distance (jaccard index, jaccard similarity coefficient)," *Dictionary of Bioinformatics and Computational Biology. 1st ed. Chichester, UK: John Wiley & Sons, Ltd*, pp. 223–270, 2004.

[58] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023. [Online]. Available: https://arxiv.org/abs/1706.03762

[59] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," 2018. [Online]. Available: https://arxiv.org/abs/1802.05365

[60] J. Herzig, T. Müller, S. Krichene, and J. M. Eisenschlos, "Open domain question answering over tables via dense retrieval," *arXiv preprint arXiv:2103.12011*, 2021.

[61] M. G. Gol, J. Pujara, and P. Szekely, "Tabular cell classification using pre-trained cell embeddings," in *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2019, pp. 230–239.

[62] Ž. Vujović *et al.*, "Classification model evaluation metrics," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 6, pp. 599–606, 2021.

[63] evidentlyai., "Accuracy," https://www.evidentlyai.com/classification-metrics/multi-class-metrics, 2024, accessed: 2024-06-06.

[64] Y. Sui, M. Zhou, M. Zhou, S. Han, and D. Zhang, "Table meets llm: Can large language models understand structured table data? a benchmark and empirical study," in *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, 2024, pp. 645–654.

[65] Y. Liu, T. Han, S. Ma, J. Zhang, Y. Yang, J. Tian, H. He, A. Li, M. He, Z. Liu *et al.*, "Summary of chatgpt-related research and perspective towards the future of large language models," *Meta-Radiology*, p. 100017, 2023.

[66] Z. Chen, S. Dadiomov, R. Wesley, G. Xiao, D. Cory, M. Cafarella, and J. Mackinlay, "Spreadsheet property detection with rule-assisted active learning," ser. CIKM. ACM, 2017.

[67] Z. Wang, H. Zhang, C. Li, J. M. Eisenschlos, V. Perot, Z. Wang, L. Miculicich, Y. Fujii, J. Shang, C. Lee, and T. Pfister, "Chain-of-table: Evolving tables in the reasoning chain for table understanding," *CoRR*, vol. abs/2401.04398, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2401.04398

[68] A. L. Gentile, P. Ristoski, S. Eckel, D. Ritze, and H. Paulheim, "Entity matching on web tables: a table embeddings approach for blocking," in *EDBT*, 2017.

[69] N. Hollmann, S. Müller, K. Eggensperger, and F. Hutter, "Tabpfn: A transformer that solves small tabular classification problems in a second," vol. 33, 2022, pp. 11 033–11 043.

[70] M. Joseph and H. Raj, "Gate: Gated additive tree ensemble for tabular classification and regression," *arXiv preprint arXiv:2207.01848*, 2022.

[71] K. Sun, H. Rayudu, and J. Pujara, "A hybrid probabilistic approach for table understanding," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 5, 2021, pp. 4366–4374.

[72] M. Ghasemi-Gol, J. Pujara, and P. Szekely, "Learning cell embeddings for understanding table layouts," *Knowledge and Information Systems*, vol. 63, no. 1, pp. 39–64, 2021.

[73] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on tabular data?" *arXiv preprint arXiv:2207.08815*, 2022.

[74] I. Rubachev, A. Alekberov, Y. Gorishniy, and A. Babenko, "Revisiting pretraining objectives for tabular deep learning," *arXiv preprint arXiv:2207.03208*, 2022.

[75] R. Levin, V. Cherepanova, A. Schwarzschild, A. Bansal, C. B. Bruss, T. Goldstein, A. G. Wilson, and M. Goldblum, "Transfer learning with deep tabular models," *arXiv preprint arXiv:2206.15306*, 2022.

[76] A. Dubey, F. Radenovic, and D. Mahajan, "Scalable interpretability via polynomials," *arXiv preprint arXiv:2205.14108*, 2022.

[77] B. Schäfl, L. Gruber, A. Bitto-Nemling, and S. Hochreiter, "Hopular: Modern hopfield networks for tabular data," *arXiv preprint arXiv:2206.00664*, 2022.

[78] F. Radenovic, A. Dubey, and D. Mahajan, "Neural basis models for interpretability," *arXiv preprint arXiv:2205.14120*, 2022.

[79] Y. Gorishniy, I. Rubachev, and A. Babenko, "On embeddings for numerical features in tabular deep learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 991–25 004, 2022.

[80] J. Chen, K. Liao, Y. Wan, D. Z. Chen, and J. Wu, "Danets: Deep abstract networks for tabular data classification and regression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 4, 2022, pp. 3930–3938.

[81] V. Borisov, T. Leemann, K. Seßler, J. Haug, M. Pawelczyk, and G. Kasneci, "Deep neural networks and tabular data: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[82] S. Cai, K. Zheng, G. Chen, H. Jagadish, B. C. Ooi, and M. Zhang, "Arm-net: Adaptive relation modeling network for structured data," in *Proceedings of the 2021 International Conference on Management of Data*. Association for Computing Machinery, 2021, pp. 635–649.

[83] V. Varik, A. Berg, P. Kokkalis, J. Yosinski, J. Dean, and V. Garg, "Distracting features in deep tabular data learning: An information-theoretic perspective," *arXiv preprint arXiv:2206.06153*, 2022.

[84] A. Bogatu, A. A. Fernandes, N. W. Paton, and N. Konstantinou, "Dataset discovery in data lakes," in *2020 ieee 36th international conference on data engineering (icde)*. IEEE, 2020, pp. 709–720.

[85] R. C. Fernandez, Z. Abedjan, F. Koko, G. Yuan, S. Madden, and M. Stonebraker, "Aurum: A data discovery system," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 2018, pp. 1001–1012.

[86] F. Nargesian, K. Q. Pu, E. Zhu, B. Ghadiri Bashardoost, and R. J. Miller, "Organizing data lakes for navigation," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 1939–1950.

[87] G. Fan, J. Wang, Y. Li, D. Zhang, and R. J. Miller, "Semantics-aware dataset discovery from data lakes with contextualized column-based representation learning," *Proceedings of the VLDB Endowment*, vol. 16, no. 7, pp. 1726–1739, 2023.