Automatic Domain Classification of Tabular Datasets Using Large Language Models

Elizaveta Gamper[1,†], Irina Deeva[1,2,*,†]

[1]ITMO University, Saint-Petersburg, Russia
[2]AI Institute, Saint-Petersburg, Russia

Abstract

In this paper, we propose a pragmatic pipeline for the automated domain classification of tabular datasets. This pipeline integrates large language models (LLMs) with an interpretable tag-clustering procedure, thereby generating dataset-level domains. This task is of paramount importance for the discovery of datasets, the enrichment of metadata, and the construction of domain-aware data assistants that assist researchers and practitioners in the more effective finding, combining, and reusing of tabular data. The proposed methodology involves the conversion of noisy dataset tags into a compact, human-readable domain palette via the utilisation of embedding and clustering techniques. The approach leverages structured dataset samples (column names with optional rows) under zero-, one-, and few-shot prompting strategies to assign domains. Empirical evaluations against classical baselines – term/tag overlap, column-name embedding classifiers, and meta-feature models – indicate that modern LLMs demonstrate competitiveness in few-shot settings, though sensitivity to input format and demonstration selection is observed. Concurrently, classical methods exhibit complementarity in certain retrieval metrics. A series of ablation studies have demonstrated that the incorporation of row samples, in conjunction with judicious demonstration choice, serves to enhance the robustness of the system. It is asserted that the present contribution consists of an interpretable domain palette, a constructed benchmark of diverse tabular datasets, and reproducible code and data. The purpose of these contributions is to enable further research on domain discovery and domain-aware tooling for tabular data.

Keywords

Tabular data, Domain classification, Large language models, Tag clustering, Few-shot learning

1. Introduction

Tabular datasets remain the dominant format for real-world data in science, engineering, and industry, yet their sheer heterogeneity makes automated understanding and discovery difficult. Nevertheless, such an analysis may possess a considerable degree of practical significance. For example, for next-generation AI research assistants - systems that must search corpora of datasets, recommend modeling pipelines, generate meaningful prompts, and explain results to human analysts - the ability to recognize the subject domain of a table automatically is a foundational capability. Domain labels enable assistants to surface relevant prior work, suggest domain-appropriate preprocessing and models, and constrain retrieval to semantically coherent collections; without them, downstream recommendations are brittle or require costly human curation.

In this paper we propose and evaluate a practical pipeline for automatic domain classification of tabular datasets using large language models (LLMs). Our approach combines two complementary ideas. First, we induce interpretable dataset-level domains by clustering natural-language tags (e.g., OpenML tags) into semantically coherent groups, producing a compact and human-readable domain palette suitable for benchmark construction and downstream use. Second, we leverage general-purpose LLMs to perform dataset-level classification directly from small samples, exploring zero-, one-, and few-shot prompting strategies and comparing LLM performance to classical embedding-and-clustering baselines. To enable rigorous evaluation we construct a benchmark of hundreds of diverse OpenML

IRCDL 2026: 22nd Conference on Information and Research Science Connecting to Digital and Library Science, February 19-20, 2026, Modena, Italy

*Corresponding author.
†These authors contributed equally.
471825@niuitmo.ru (E. Gamper); iriny.deeva@gmail.com (I. Deeva)

datasets mapped to a 14-domain taxonomy, and we provide controlled experiments and ablations that probe sensitivity to sample size, shot selection, and input formats.

Contributions. Summarized briefly, this paper (1) presents a tag-clustering method to produce an interpretable domain palette and an accompanying benchmark of OpenML tables, and (2) evaluates LLM-based classification strategies (zero/one/few-shot) and ablations against classical baselines, documenting where LLMs help and where they remain brittle. Together, these elements advance practical, reproducible methods for building domain-aware assistant behaviors over tabular data. The benchmark and experiments code are available in the repository https://github.com/ITMO-NSS-team/TabDomain_Extractor.

2. Related works

2.1. Tabular data domain discovery

Tabular data domain discovery was motivated by the rising popularity of data lakes and repositories that contained large-scale unstructured tables that required automatic identification of semantics to be able to search, integrate and analyze them. Methods for discovery of domains have evolved in several phases.

Rule-based and dictionary-driven approaches. Early methodologies for table understanding approached via handcrafted rules, dictionaries, and regular expressions to characterize column semantics (e.g., detecting ZIP codes, dates). These methods documented in earlier work [1] on data integration and schema matching, were straightforward and interpretable, though they lacked robustness: they did not handle data noise or heterogeneity, and they did not allow the integration at a higher level of abstraction - beyond a single table.

Column-level semantic typing with supervised learning. Machine learning was later included to classify individual columns in the next phase. A key benchmark in this direction is Sherlock [2] that integrated embeddings of cell values, headers of columns, and character-level features into a deep semantic type model. Later extensions achieved even higher accuracy by utilizing metadata and table context. Although these models outperformed rule-based baselines, they were limited by hierarchical taxonomies of semantic types, and still worked at the level of columns rather than dataset-level domains.

Domain discovery based on corpus-level and embedding methods. Researchers proposed corpus-level approaches to advance beyond column classification. The framework D4 for discovering latent domains by learning contextual embeddings of column values and clustering them across large corpora was introduced in [3]. With this shift, the focus of the problem changed from annotating individual columns to revealing domains of datasets. Nonetheless, methods based on embeddings depend on co-occurrence statistics and stable lexical anchors (such as column names and frequent terms). Consequently, their performance diminishes on small, heterogeneous collections characterized by sparse embeddings.

Hybrid and specialized systems. More recent work has sought to bridge the gap between column-level annotation and holistic discovery. Sato[4] combined neural models with table context features, resulting in robustness on messy enterprise data. Pythagoras[5] turned its attention to the overlooked difficulty of numerical columns, demonstrating that numerous classifiers based on textual attributes do not perform well when numbers take precedence. Practical annotation tool Kepler-aSI[6] combined machine learning, knowledge-graph lookups, and human-in-the-loop annotation to create semantic descriptions of datasets. Although these systems are useful in practice, they often necessitate user interaction or external knowledge bases, hindering fully automatic, fine-grained domain discovery on a large scale.

2.2. LLM for tabular data analysis

To address the limitations of classical tabular domain discovery, LLMs offer essential capabilities. They can generalize beyond rigid taxonomies, manage datasets with both numbers and text, and enable reasoning at the dataset level, even with small or varied collections. Consequently, LLMs are a natural

fit for automated, scalable domain classification, driving our research into using them for dataset-level tabular comprehension.

Table-specific pretraining Transformers. An impactful line of investigation involves the adaptation of pretrained Transformer models for structured tabular data. TaBERT[7] learned representations of tables and natural language together, facilitating question answering and semantic parsing on relational data. TAPAS[8] built upon BERT by linearizing tables and using weak supervision from question-answer pairs during pretraining, showing impressive results on table QA benchmarks. TAPEX[9] advanced this concept further by generating SQL queries and training models to estimate their execution, which resulted in considerable improvements in reasoning tasks. The models demonstrated that pretraining language models can be tailored for tables, but their objectives are specific to tasks (such as QA and text-to-SQL) rather than aimed at higher-level dataset discovery or domain classification.

General-purpose LLMs with prompting and in-context learning. As large, general-purpose language models emerged, researchers started investigating their capacity to comprehend tabular data without task-specific pretraining. The research [10] has systematically assessed GPT-style models on table reasoning tasks, demonstrating that although LLMs can extract and reason about table content, they are very sensitive to the format and size of the table. Robustness was enhanced by new prompting strategies. Chain-of-Table[11] broke down table reasoning into intermediate steps, whereas Automatic Demonstration Selection[12] focused on optimizing in-context examples for table tasks. Other studies (e.g., [13]) investigated the direct application of GPT models for generating summaries or SQL queries based on tabular data. Even with their encouraging outcomes, these methods present practical difficulties: their effectiveness is greatly influenced by the design of prompts, demonstrations, and inference expenses. Furthermore, they concentrate on single-table reasoning rather than multi-dataset domain discovery.

Surveys and perspectives. Recent surveys have consolidated this quickly developing field. According to [14], LLM applications in tabular data encompass prediction, generation, QA, and understanding. They highlight advantages like flexibility and natural language alignment, but also point out drawbacks such as instability and numerical brittleness. Authors of [15] also concentrate on the table reasoning abilities of LLMs, assessing their benefits compared to pre-LLM models and emphasizing unresolved issues. In [16] the researches complement this by surveying dataset discovery in the LLM era, highlighting opportunities for domain inference while stressing that system-level integration remains largely unresolved.

3. Proposed approach

3.1. Forming domains via tag clustering

In order to ascertain the capacity of LLM to determine the domain of a tabular dataset, it is necessary to establish an appropriate benchmark that reflects a sufficiently diverse range of domains. We construct a benchmark for domain recognition of tabular datasets as follows. Let $\mathcal{D} = \{D_1, \ldots, D_N\}$ be a collection of datasets, each associated with a set of tags $\mathcal{T}_i = \{t_{i1}, \ldots, t_{im_i}\}$ from a vocabulary $\mathcal{V}$. We fix a tag embedding function $\phi : \mathcal{V} \to \mathbb{R}^p$ and obtain a dataset-level embedding by aggregating the tag embeddings, e.g. by averaging:

$$z_i = \frac{1}{m_i} \sum_{j=1}^{m_i} \phi(t_{ij}) \in \mathbb{R}^p.$$

Collecting z_i into a matrix $Z \in \mathbb{R}^{N \times p}$, we apply a clustering algorithm $\mathcal{C} : \mathbb{R}^{N \times p} \to \{1, \ldots, K\}^N$ to obtain cluster assignments $y_i = \mathcal{C}(Z)_i$, which we interpret as domain labels. Thus domains correspond to clusters $\mathcal{D}_k = \{D_i \in \mathcal{D} \mid y_i = k\}$, and we define a supervised benchmark with input space $\mathcal{X}$ (representations of tabular datasets), label space $\mathcal{Y} = \{1, \ldots, K\}$, and ground-truth labeling function $h^\star : \mathcal{D} \to \mathcal{Y}$ given by $h^\star(D_i) = y_i$. In our setting, *tags* are short human-provided textual labels used in OpenML[1] to annotate datasets, tasks, and benchmarks. They serve as lightweight semantic metadata

[1] https://www.openml.org/

that describe, for example, the application area (e.g., `bioinformatics`, `finance`), data modality or structure (e.g., `text`, `multilabel`), and benchmark or challenge membership (e.g., `OpenML100`). OpenML users attach such tags to resources to facilitate organization and search; we treat these user-defined tags as noisy but informative descriptors of the underlying domain of each tabular dataset. Although OpenML tags are user-generated and potentially noisy, our pipeline relieves this through normalization, semantic embedding, clustering, and manual domain interpretation. Any residual noise affects all evaluated methods equally, preserving the validity of comparative results.

3.2. LLM for tabular data domain discovery

We consider a benchmark of tabular datasets $\mathcal{D} = \{D_1, \ldots, D_N\}$ with associated domain labels from a finite set $\mathcal{Y} = \{c_1, \ldots, c_K\}$, obtained as described in the previous section. The benchmark is partitioned into a training index set $\mathcal{I}_{\text{train}}$ and a test index set $\mathcal{I}_{\text{test}}$, but the training portion is used only to provide in-context examples rather than to update model parameters. Each dataset D_i is represented to the LLM via a subsampling operator r that selects a finite collection of rows $X_i = r(D_i)$ (e.g., a few dozen rows of the table). For a given LLM with parameters θ, we treat it as a black-box conditional model F_θ over textual outputs given a textual prompt P, inducing a distribution $F_\theta(\cdot \mid P)$ over completions. To query the model on a test dataset D_i, we construct a prompt

$$P_i = \Pi\big(\mathcal{Y},\, S,\, X_i\big),$$

where Π is a deterministic prompt-construction function that concatenates (i) a natural-language task description ("classify the domain of the dataset"), (ii) the list of possible domains $\mathcal{Y}$, (iii) a set $S \subseteq \{(X_j, y_j) : j \in \mathcal{I}_{\text{train}}\}$ of example pairs (empty in the zero-shot regime, $|S| = 1$ in the one-shot regime, and $|S| = m$ for few-shot, typically $m = 5$), and (iv) the representation X_i of the test dataset (see figure 1).

The LLM is instructed to return a structured textual object (JSON) encoding an ordered tuple of domain predictions

$$\hat{\mathbf{y}}_i = \big(\hat{y}_i^{(1)}, \hat{y}_i^{(2)}, \hat{y}_i^{(3)}\big), \quad \hat{y}_i^{(\ell)} \in \mathcal{Y} \cup \{\varnothing\},$$

where $\hat{y}_i^{(1)}$ denotes the main domain and $\hat{y}_i^{(2)}, \hat{y}_i^{(3)}$ are (optionally) up to two auxiliary domains; $\varnothing$ indicates the absence of an auxiliary label. Operationally, we obtain $\hat{\mathbf{y}}_i$ by applying a deterministic parser Γ to a single sample from $F_\theta(\cdot \mid P_i)$: $\hat{\mathbf{y}}_i = \Gamma\big(F_\theta(\cdot \mid P_i)\big)$. Zero-shot, one-shot, and few-shot experiments differ only in the choice of S in the prompt construction; evaluation then compares $\hat{\mathbf{y}}_i$ with the ground-truth domain labels y_i for $i \in \mathcal{I}_{\text{test}}$ under appropriate multi-label metrics.

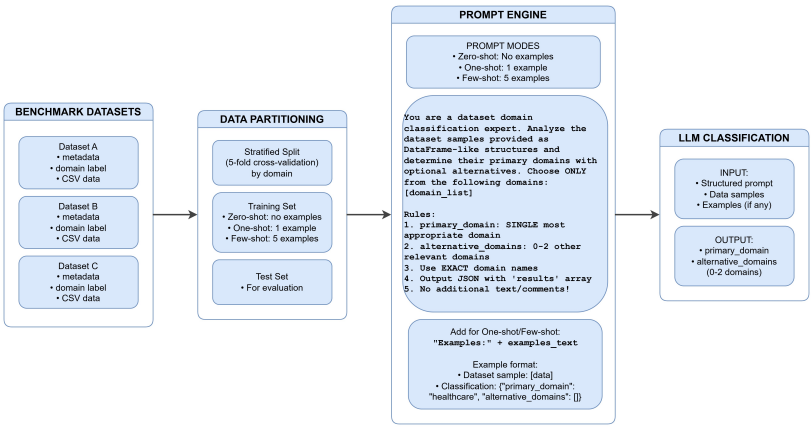

Figure 1: The schema of proposed approach for tabular data domain discovery with LLM

4. Experiments design

4.1. Benchmark construction

This experiment aims to construct a comprehensive benchmark of tabular datasets for domain classification and, at the same time, to systematically study which combinations of tag-embedding constructions and clustering algorithms are best suited for the underlying domain-induction task. We collect datasets $\{D_1, \ldots, D_N\}$ from OpenML, prioritizing those annotated with user-provided tags in order to maximize semantic coverage and diversity of application areas. For each dataset we retrieve and store the full table together with its metadata (ID, name, tags, feature descriptors, number of rows). All tags are aggregated into a global vocabulary and normalized: tags are lowercased, special characters and stop words are removed, and noisy tags that cannot meaningfully act as domains (e.g., names of benchmarks or hosting services) are filtered out.

To induce domains, we embed the cleaned tag vocabulary into a vector space using three embedding constructions: Sentence Transformers (all-mpnet-base-v2) [17], TF–IDF [18], and Word2Vec [19]. On top of each embedding space we apply a suite of clustering algorithms—HDBSCAN [20], k-means [21], agglomerative clustering [22], and DBSCAN [23]—with fixed hyperparameters (e.g., minimum cluster size, distance threshold, number of clusters), thereby generating multiple candidate partitions of the tag space. For each embedding–clustering configuration we compute internal validity indices, namely the Silhouette score [24] and the Davies–Bouldin score [25], to assess cluster compactness and separation and to identify the most suitable configurations for benchmark construction. Clusters of semantically similar tags are then manually interpreted as domains (e.g., a cluster containing `medical`, `health` is labeled `medicine`) and saved in a machine-readable mapping. Finally, each dataset D_i is assigned to the domain with which it has the largest overlap in tags, defaulting to an `Other` domain if no overlap exists; domains with fewer than a target number of datasets (e.g., 5) are enriched by querying OpenML for additional datasets whose tags match the domain vocabulary, yielding a balanced and well-structured benchmark that also encodes the empirical preferences over embedding and clustering choices.

4.2. LLM for tabular data domain discovery

This subsection describes the main stage of the classification experiments, which assess the capacity of large language models (LLMs) to autonomously identify the domain of tabular datasets in zero-shot, one-shot, and few-shot regimes. The goal is twofold: (i) to quantify how well LLMs can generalize to unseen datasets in a realistic open-domain setting where only a small number of examples can be provided due to context-length constraints, and (ii) to compare their performance against traditional, non-LLM baselines under identical evaluation protocols.

Experimental setup and execution. We consider the benchmark collection of tabular datasets $\{D_1, \ldots, D_N\}$ with domain labels $y_i \in \mathcal{Y}$ constructed as described in the previous subsection. For each LLM (DeepSeek-V3 and `gpt-4o`), and for each in-context regime $m \in \{0, 1, 5\}$ (zero-shot, one-shot, few-shot), we perform 5 independent stratified splits of the dataset indices into training and test sets $(\mathcal{I}^{(s)}_{\text{train}}, \mathcal{I}^{(s)}_{\text{test}})$, $s = 1, \ldots, 5$, with $|\mathcal{I}^{(s)}_{\text{test}}|/N = 0.2$ and stratification preserving the empirical domain distribution. Each dataset D_i is mapped to a compact tabular representation $X_i = r(D_i)$ by a deterministic subsampling operator r that selects all column headers and the first 5 data rows, formatted as a structured table (e.g., Markdown-like). For each split s and regime m, we sample a set of labeled training examples

$$S^{(s,m)} \subseteq \{(X_j, y_j) : j \in \mathcal{I}^{(s)}_{\text{train}}\},$$

with $|S^{(s,0)}| = 0$ (zero-shot), $|S^{(s,1)}| = 1$ (one-shot), and $|S^{(s,5)}| = 5$ (few-shot), choosing the elements of $S^{(s,m)}$ uniformly at random. For each test dataset D_i with $i \in \mathcal{I}^{(s)}_{\text{test}}$ we construct a textual prompt

$$P^{(s,m)}_i = \Pi\big(\mathcal{Y},\, S^{(s,m)},\, X_i\big),$$

where Π is a fixed prompt-construction function that concatenates: (i) a natural-language task description instructing the model to classify the domain of the given dataset, (ii) the list of valid domains $\mathcal{Y}$,

(iii) the example pairs in $S^{(s,m)}$ (omitted in zero-shot), and (iv) the representation X_i of the test dataset. The LLM is queried as a black-box conditional generator $F_\theta(\cdot \mid P)$ and is instructed to respond with a JSON object specifying an ordered list of up to three domain labels. A deterministic parser Γ is applied to the generated text to extract an ordered prediction tuple

$$\hat{\mathbf{y}}_i^{(s,m)} = \big(\hat{y}_i^{(s,m,1)}, \hat{y}_i^{(s,m,2)}, \hat{y}_i^{(s,m,3)}\big), \quad \hat{y}_i^{(s,m,\ell)} \in \mathcal{Y} \cup \{\varnothing\},$$

where $\hat{y}_i^{(s,m,1)}$ is treated as the primary domain prediction and $\hat{y}_i^{(s,m,2)}, \hat{y}_i^{(s,m,3)}$ as optional alternatives ($\varnothing$ denotes "no label provided"). All experimental runs use shared decoding settings (e.g., temperature = 0.3, maximum tokens = 500) fixed across models and modes; prompts, random splits, and example selections are held constant across LLMs to enable fair comparison.

Model access. All LLM inferences were performed via the OpenRouter API. The following specific model endpoints were used: openai/gpt-4o for gpt-4o and deepseek/deepseek-chat for DeepSeek-V3. This unified access point ensured consistent query formatting and billing.

Evaluation metrics. The performance of each LLM, for each regime $m \in \{0, 1, 5\}$, is evaluated on the test sets $\mathcal{I}_{\text{test}}^{(s)}$ using three standard metrics. (i) *Accuracy* is computed as the proportion of test datasets for which the primary prediction matches the ground-truth domain,

$$\text{Acc}^{(m)} = \frac{1}{5}\sum_{s=1}^{5} \frac{1}{|\mathcal{I}_{\text{test}}^{(s)}|} \sum_{i \in \mathcal{I}_{\text{test}}^{(s)}} \mathbf{1}\{\hat{y}_i^{(s,m,1)} = y_i\}.$$

(ii) *Top-3 Recall* measures the proportion of test datasets for which the true domain appears in any of the up to three predicted labels,

$$\text{Top3Rec}^{(m)} = \frac{1}{5}\sum_{s=1}^{5} \frac{1}{|\mathcal{I}_{\text{test}}^{(s)}|} \sum_{i \in \mathcal{I}_{\text{test}}^{(s)}} \mathbf{1}\{y_i \in \{\hat{y}_i^{(s,m,1)}, \hat{y}_i^{(s,m,2)}, \hat{y}_i^{(s,m,3)}\}\}.$$

(iii) *Macro F1-score* is computed by first deriving per-domain precision and recall from the primary predictions, then averaging the F1-scores uniformly over all domains, which makes the metric sensitive to performance on rare domains. For each metric, we aggregate results across the 5 splits and report the mean and standard deviation.

Comparison with baseline methods. To contextualize LLM performance, we evaluate three non-LLM baselines on exactly the same 5 stratified train/test splits. (1) *Data-driven domain discovery* [26]: an unsupervised method that extracts domain-specific terms from column values (discretizing numeric columns via `pd.qcut`), groups terms into equivalence classes, constructs local and strong domains using Jaccard similarity and graph connectivity, and classifies test datasets by maximum overlap with domain-specific terms after filtering terms that appear in more than 80% of domains. (2) *Embeddings of columns + supervised classification*: column names are embedded using the all-MiniLM-L6-v2 sentence transformer, and domains are predicted either by a per-column nearest-neighbor voting scheme (using FAISS for similarity search) or by averaging embeddings per dataset and training four supervised classifiers (Random Forest, Logistic Regression, SVM, KNN) with stratified 5-fold cross-validation and class-balanced weights. (3) *Meta-feature-based classification*: statistical and information-theoretic meta-features are extracted via the `pymfe` library [27]; features with more than 50% missing values, constant features, and duplicates are removed; multicollinearity is controlled via a variance inflation factor (VIF) threshold < 10; and feature selection is performed using a PyTorch Lasso model. The same four classifiers as above are then trained with stratified 5-fold cross-validation, and the best-performing model per configuration is retrained on the full training data for final evaluation on the held-out test sets. All baselines use the same label space $\mathcal{Y}$ and evaluation metrics as the LLMs, ensuring a controlled comparison.

4.3. Ablation studies

To investigate the impact of input data representation on LLM performance, we conducted a series of ablation experiments that vary only the way each tabular dataset is rendered in the prompt, while keeping all other components of the protocol fixed (same LLMs, same domain label space $\mathcal{Y}$, same train/test splits, prompting strategy, and evaluation metrics). The central question is whether LLMs primarily rely on column semantics (names) or also benefit significantly from access to cell-level values.

Let $\{D_1, \ldots, D_N\}$ and the corresponding domain labels $\{y_1, \ldots, y_N\}$ be the benchmark constructed as in the previous subsections, and let $(\mathcal{I}^{(s)}_{\text{train}}, \mathcal{I}^{(s)}_{\text{test}})$, $s = 1, \ldots, 5$, denote the same stratified splits used in the main LLM experiments, with a test fraction of 20%. For each dataset D_i we define two deterministic representation operators:

- **Headers-only representation** $r^{(H)}$: $X_i^{(H)} = r^{(H)}(D_i)$ is a textual object consisting only of the list of column names, formatted as a simple schema (e.g., a one-line or bullet-list enumeration of headers).
- **Tabular-rows representation** $r^{(T)}$: $X_i^{(T)} = r^{(T)}(D_i)$ is a structured table containing all column names and up to 50 data rows, obtained by uniformly sampling up to 50 rows from D_i without replacement (if D_i has fewer than 50 rows, all rows are used).

For each in-context regime $m \in \{0, 1, 5\}$ (zero-shot, one-shot, few-shot), and for each split s, we reuse the same sets of labeled examples $S^{(s,m)} \subseteq \{(X_j, y_j) : j \in \mathcal{I}^{(s)}_{\text{train}}\}$ as in the main experiments, where $|S^{(s,0)}| = 0$, $|S^{(s,1)}| = 1$, and $|S^{(s,5)}| = 5$. To ensure strict comparability, the example indices and their associated labels are identical across ablations; only the representation X_j is swapped between $X_j^{(H)}$ and $X_j^{(T)}$ depending on the ablation condition.

For each representation type $r^{(\cdot)} \in \{r^{(H)}, r^{(T)}\}$, split s, regime m, and test dataset D_i with $i \in \mathcal{I}^{(s)}_{\text{test}}$, we construct the prompt

$$P_i^{(s,m,(\cdot))} = \Pi\big(\mathcal{Y},\, S^{(s,m,(\cdot))},\, X_i^{(\cdot)}\big),$$

where Π is the same prompt-construction function used in the main LLM experiments and $S^{(s,m,(\cdot))}$ is obtained from $S^{(s,m)}$ by replacing each X_j with $X_j^{(\cdot)}$. Thus, the system instructions, domain list, and example set structure are identical across ablations; only the content of the dataset representations changes. The LLM is queried with $P_i^{(s,m,(\cdot))}$ using the same decoding parameters (temperature, maximum length, etc.), and the output is parsed via the same deterministic parser Γ to obtain the ordered prediction tuple

$$\hat{\mathbf{y}}_i^{(s,m,(\cdot))} = \big(\hat{y}_i^{(s,m,(\cdot),1)}, \hat{y}_i^{(s,m,(\cdot),2)}, \hat{y}_i^{(s,m,(\cdot),3)}\big).$$

Both ablation conditions are evaluated under exactly the same protocol as the main LLM experiments: we compute Accuracy (primary label), Top-3 Recall (primary + up to two alternatives), and macro F1-score per split, and then aggregate results across the 5 stratified splits by reporting the mean and standard deviation for each metric, representation type, LLM, and in-context regime. This design isolates the effect of input representation, enabling a direct comparison of how much additional benefit, if any, is gained by providing cell-level tabular values beyond column headers alone.

5. Experimental results

5.1. Benchmark construction

The initial data collection involved sourcing roughly 100 tagged tabular datasets from OpenML. Each dataset was downloaded as a `pandas` DataFrame and stored as `full_dataset.csv` in a dedicated directory, while the associated metadata (dataset ID, name, tags, feature descriptions, and row count) was serialized to `metadata.json`. All tags attached to these datasets were aggregated into a global

vocabulary and saved in `unique_tags.json`, forming the basis for subsequent embedding and clustering steps. This collection constitutes the raw pool from which domains are induced and to which domain labels are later assigned.

Table 1 summarizes the results of combining three embedding constructions (Sentence Transformers, TF–IDF, Word2Vec) with four clustering algorithms (HDBSCAN, k-means, agglomerative clustering, DBSCAN), evaluated using the Silhouette score ("Silh") and the Davies–Bouldin index ("DB"). While TF–IDF with agglomerative clustering achieves slightly better internal validity scores numerically (Silh $= 0.131$, DB $= 1.749$) than Sentence Transformers with agglomerative clustering (Silh $= 0.106$, DB $= 1.781$), the latter configuration produced clusters that were substantially more coherent and interpretable in terms of high-level semantic domains (e.g., clearer separation between technical, biomedical, and financial topics). Methods that yielded only a very small number of clusters (e.g., DBSCAN with Sentence Transformer embeddings, producing two almost trivial clusters) or exhibited poor separation (high DB index) were discarded. Based on this combination of quantitative criteria and manual inspection of tag groupings, we selected agglomerative clustering in the Sentence Transformer embedding space as the basis for defining the final set of domains.

Table 1
Results of different clusterization methods using various embedding methods

Embedding	Clusterization	Clusters	Silh	DB
ST	HDBSCAN	8	0.052	2.675
ST	KMEANS	8	0.044	2.897
ST	AGGLOMERATIVE	14	0.106	1.781
ST	DBSCAN	2	0.931	3.005
TFIDF	HDBSCAN	8	0.040	4.127
TFIDF	KMEANS	8	0.073	1.914
TFIDF	AGGLOMERATIVE	15	0.131	1.749
TFIDF	DBSCAN	2	0.025	1.699
WORD2VEC	HDBSCAN	5	0.014	4.062
WORD2VEC	KMEANS	6	0.010	3.640
WORD2VEC	AGGLOMERATIVE	15	0.016	2.414
WORD2VEC	DBSCAN	4	0.021	4.126

Dataset–Domain Mapping and Enrichment. The chosen clustering of tag embeddings induces 14 semantic domains, which were manually named by inspecting the tags in each cluster (e.g., a cluster containing `medical`, `health`, `patient` was labeled *Medical Science*). Each dataset was then assigned to the domain with which it has the largest overlap in cleaned tags, defaulting to an *Other*-type domain when no overlap occurred; small, semantically redundant clusters were merged during this process. To avoid severely underrepresented domains, clusters with fewer than five datasets were enriched by querying OpenML for additional datasets whose tags matched the corresponding domain's tag set, selecting up to ten additional datasets per domain according to tag-overlap scores. The final benchmark, after this enrichment step, comprises 258 tabular datasets distributed across 14 domains, each dataset accompanied by full data, standardized samples, and metadata, thereby ensuring diversity and scalability in the context of tabular data. The resulting domain distribution is illustrated in the pie chart in figure 2, showing that domains such as *Computer Science & Artificial Intelligence*, *Medical Science*, and *Engineering & Technology* are most frequent, while highly specialized areas like *Veterinary Science* remain comparatively small but still represented. Representative word clouds for several domains (figure 3) qualitatively confirm that the induced domains capture coherent thematic groupings of tags, supporting their use as ground-truth labels for subsequent domain-classification experiments.

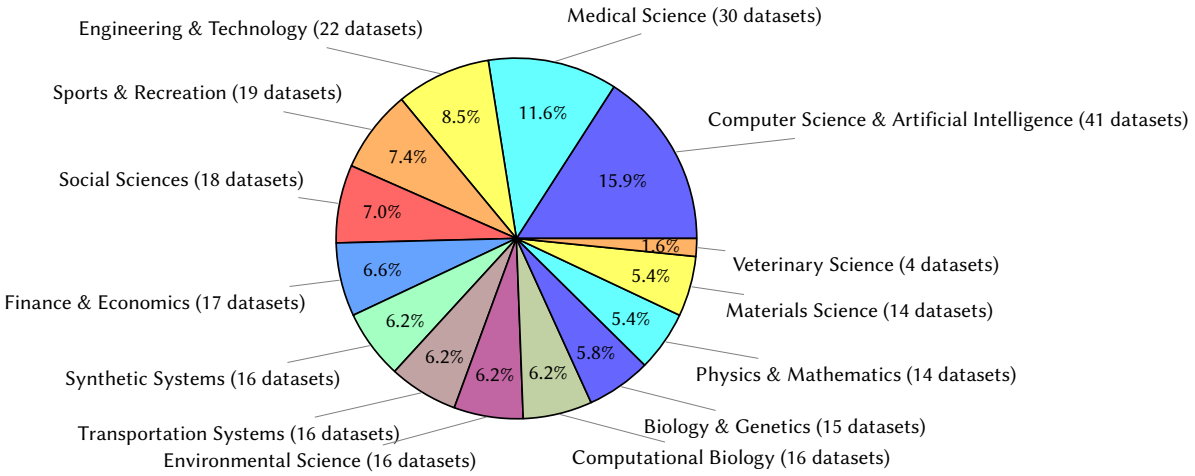

Figure 2: Domains distribution in a final benchmark

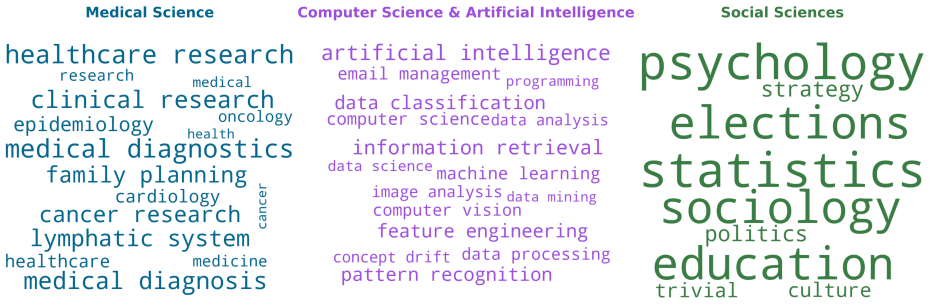

Figure 3: Word clouds of tags for some domains

5.2. LLM for tabular data domain discovery

Table 2 summarizes the overall performance of `gpt-4o` and DeepSeek-V3 across zero-shot, one-shot and few-shot regimes on the full benchmark. Both models already achieve non-trivial accuracy in the zero-shot setting (0.61 for `gpt-4o` and 0.55 for DeepSeek-V3), confirming that the models can often infer the domain purely from schema and a small sample of rows. Moving from zero-shot to few-shot consistently improves performance for both models: `gpt-4o` reaches 0.63 accuracy and 0.58 macro F1-score in the few-shot regime, while DeepSeek-V3 attains 0.62 accuracy and 0.55 F1. The gains are especially pronounced for DeepSeek-V3, whose Top-3 recall improves from 0.72 to 0.81 when moving from zero-shot to few-shot, indicating that additional in-context examples help the model better calibrate its ranking over candidate domains. For `gpt-4o`, the few-shot regime yields the best accuracy and F1, but Top-3 recall slightly decreases relative to the zero-shot setting; this suggests that the model becomes more decisive (placing more probability mass on a single domain), which improves top-1 accuracy but reduces the diversity of secondary predictions.

A comparison with non-LLM baselines is given in Table 3. The best classical method based on sentence embeddings of column names achieves 0.61 accuracy and 0.59 macro F1, while meta-feature-based classification and the D4 domain-discovery method lag behind (0.44 and 0.27 accuracy, respectively). The few-shot `gpt-4o` configuration attains slightly higher accuracy (0.63) than the embedding-based baseline and comparable macro F1 (0.58), despite not being trained on this benchmark and only seeing a handful of in-context examples. At the same time, the embedding baseline has a noticeably higher Top-3 recall (0.82 vs. 0.63), reflecting that it is more effective at including the correct domain somewhere in the top-ranked candidates, whereas the LLM tends to produce sharper, more concentrated predictions. Overall, these results indicate that a modern LLM in a few-shot setting is competitive with specialized

feature-engineering pipelines, and clearly dominates meta-feature and D4 approaches on this task.

The ablation where only column names are provided to the models is reported in Table 4. Interestingly, few-shot performance with headers alone is very close to, and in some metrics slightly better than, the configuration that also exposes up to 50 rows: for `gpt-4o`, few-shot accuracy increases from 0.63 to 0.64 and macro F1 from 0.58 to 0.60 when restricting the input to column headers, and DeepSeek-V3 exhibits a similar trend (few-shot accuracy 0.62 and F1 0.56). This finding can be attributed to several factors. First, column names (headers) often contain rich, domain-specific terminology that serves as a strong, condensed signal for domain inference. Second, row data can introduce noise: numeric values may be generic, categorical values may be encoded, and outliers or missing values can mislead the model's attention. Third, given the context window limitations, including many rows may dilute the presence of the most informative headers in the prompt. Finally, LLMs pretrained on vast corpora have likely developed strong priors for mapping common column name patterns to domains, whereas interpreting the semantics of raw cell values in isolation is a more ambiguous task. Thus, for the specific task of tabular datasets domain classification a concise schema representation appears to be both sufficient and optimal.

A complementary ablation varying the number of observed rows (5 vs. 50) is visualized in Figure 4. Across both models and all prompting regimes, the bars corresponding to 5-row samples are generally slightly higher than or comparable to those for 50-row samples, particularly in terms of F1-score. This pattern holds for zero-shot and few-shot settings, and is more pronounced for DeepSeek-V3. The result supports the hypothesis that small, representative samples are sufficient for LLM-based domain inference, while longer tables mostly add verbosity and potentially noisy details, especially given the limited amount of context that can be processed reliably.

Finally, Tables 5 and 6 provide a per-domain breakdown of the few-shot configuration with 50 random rows. Both models perform strongly on domains with distinctive terminology and richer training support, such as *Medical Science*, *Finance & Economics*, *Environmental Science*, *Social Sciences*, and *Transportation Systems*, where F1-scores often exceed 0.8. In contrast, performance is poor on underrepresented or semantically overlapping domains like *Biology & Genetics*, *Synthetic Systems*, and *Veterinary Science*, where F1-scores collapse to zero because the models effectively never predict these labels. The *Computer Science & Artificial Intelligence* domain also remains challenging for `gpt-4o`, with an F1-score close to zero, whereas DeepSeek-V3 achieves a modest 0.40 F1. These patterns highlight that LLM-based domain discovery is sensitive both to class imbalance and to the degree of semantic distinctiveness between domains. They also indicate where future benchmark extensions (e.g., adding more datasets or refining domain definitions) may be most beneficial for a more uniform evaluation across scientific areas.

Table 2
Classification results with gpt-4o and DeepSeek-V3 for Zero-shot, One-shot and Few-shot modes

Metric	**gpt-4o**		
	Zero-shot	**One-shot**	**Few-shot**
Accuracy	0.61 ± 0.03	0.57 ± 0.01	$\mathbf{0.63 \pm 0.01}$
Top3-recall	$\mathbf{0.72 \pm 0.04}$	0.59 ± 0.01	0.63 ± 0.01
F1-score	0.57 ± 0.05	0.52 ± 0.01	$\mathbf{0.58 \pm 0.01}$
Metric	**DeepSeek-V3**		
	Zero-shot	**One-shot**	**Few-shot**
Accuracy	0.55 ± 0.05	0.55 ± 0.01	$\mathbf{0.62 \pm 0.02}$
Top3-recall	0.72 ± 0.10	0.72 ± 0.01	$\mathbf{0.81 \pm 0.03}$
F1-score	0.47 ± 0.04	0.47 ± 0.01	$\mathbf{0.55 \pm 0.03}$

Table 3
Comparing the results of the best LLM experiment with analog algorithms

Metric	Algorithm			
	Embeddings of column names	**Meta-characteristics**	**D4**	**gpt-4o (Few-shot)**
Accuracy	0.61 ± 0.04	0.44 ± 0.03	0.27 ± 0.06	$\mathbf{0.63 \pm 0.01}$
Top-3 Recall	$\mathbf{0.82 \pm 0.02}$	0.71 ± 0.00	0.43 ± 0.07	0.63 ± 0.01
F1-score	$\mathbf{0.59 \pm 0.04}$	0.41 ± 0.02	0.24 ± 0.07	$\mathbf{0.58 \pm 0.01}$

Table 4
Column names of tabular datasets classification results with DeepSeek-V3 for Zero-shot, One-shot and Few-shot modes

Metric	gpt-4o		
	Zero-shot	**One-shot**	**Few-shot**
Accuracy	0.54 ± 0.06	0.59 ± 0.01	$\mathbf{0.64 \pm 0.01}$
Top-3 Recall	$\mathbf{0.72 \pm 0.07}$	0.61 ± 0.01	0.66 ± 0.01
F1-score	0.46 ± 0.05	0.52 ± 0.02	$\mathbf{0.60 \pm 0.02}$
Metric	**DeepSeek-V3**		
	Zero-shot	**One-shot**	**Few-shot**
Accuracy	0.53 ± 0.08	0.56 ± 0.01	$\mathbf{0.62 \pm 0.01}$
Top-3 Recall	$\mathbf{0.71 \pm 0.08}$	0.59 ± 0.01	0.66 ± 0.01
F1-score	0.43 ± 0.02	0.50 ± 0.01	$\mathbf{0.56 \pm 0.02}$

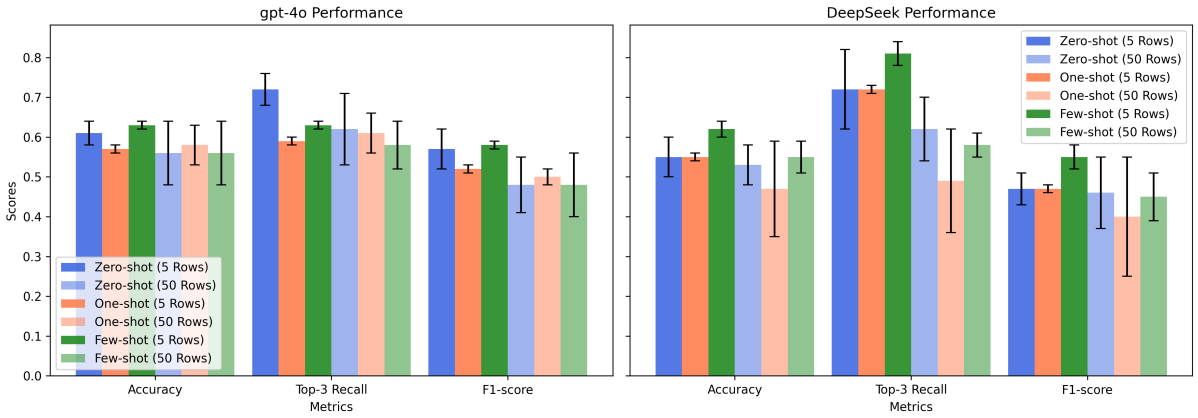

Figure 4: Comparison of gpt-4o and DeepSeek-V3 performance for 5 and 50 rows across Zero-shot, One-shot, and Few-shot modes.

6. Conclusion

In this work we introduced a practical pipeline for automatic domain classification of tabular datasets that combines interpretable tag-based domain induction with LLM-based dataset classification. Starting from noisy OpenML tags, we constructed a 14-domain palette via embedding-and-clustering, then enriched each domain with additional datasets to obtain a balanced benchmark of 258 tables spanning diverse scientific and industrial areas. The cluster analysis showed that agglomerative clustering over Sentence Transformer embeddings yields a favourable trade-off between internal validity and human interpretability, providing domain labels that are both semantically coherent and suitable as ground truth for downstream evaluation. On this benchmark we systematically evaluated two general-purpose LLMs, `gpt-4o` and DeepSeek-V3, under zero-shot, one-shot, and few-shot prompting regimes, and

Table 5
Few-Shot Classification Report for 50 random rows of tabular datasets with gpt-4o

Domain	Precision	Recall	F1-Score	Support
biology & genetics	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	8
computational biology	0.67 ± 0.47	0.67 ± 0.47	0.67 ± 0.47	2
computer science & artificial intelligence	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	16
engineering & technology	0.40 ± 0.49	0.40 ± 0.49	0.40 ± 0.49	5
environmental science	0.80 ± 0.40	0.60 ± 0.37	0.67 ± 0.37	9
finance & economics	0.77 ± 0.20	1.00 ± 0.00	0.85 ± 0.13	9
materials science	0.67 ± 0.47	0.50 ± 0.41	0.56 ± 0.42	4
medical science	0.92 ± 0.10	0.80 ± 0.18	0.84 ± 0.10	24
physics & mathematics	0.09 ± 0.11	0.40 ± 0.49	0.15 ± 0.18	8
social sciences	0.44 ± 0.13	0.87 ± 0.27	0.56 ± 0.16	13
sports & recreation	0.61 ± 0.21	0.67 ± 0.30	0.57 ± 0.14	12
synthetic systems	0.63 ± 0.45	0.40 ± 0.20	0.45 ± 0.28	10
transportation systems	0.87 ± 0.16	1.00 ± 0.00	0.92 ± 0.10	10
veterinary science	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	3

Table 6
Few-Shot Classification Report for 50 random rows of tabular datasets with DeepSeek-V3

Domain	Precision	Recall	F1-Score	Support
biology & genetics	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	8
computational biology	0.50 ± 0.50	0.50 ± 0.50	0.50 ± 0.50	2
computer science & artificial intelligence	0.31 ± 0.10	0.60 ± 0.10	0.40 ± 0.10	20
engineering & technology	0.20 ± 0.24	0.23 ± 0.29	0.21 ± 0.26	14
environmental science	1.00 ± 0.00	0.70 ± 0.24	0.80 ± 0.16	10
finance & economics	0.77 ± 0.20	0.90 ± 0.20	0.82 ± 0.18	10
materials science	0.25 ± 0.21	0.50 ± 0.45	0.33 ± 0.28	9
medical science	0.92 ± 0.10	0.84 ± 0.15	0.87 ± 0.09	24
physics & mathematics	0.30 ± 0.40	0.40 ± 0.49	0.33 ± 0.42	8
social sciences	0.59 ± 0.22	0.87 ± 0.16	0.69 ± 0.19	13
sports & recreation	0.70 ± 0.40	0.40 ± 0.25	0.50 ± 0.30	15
synthetic systems	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	10
transportation systems	0.63 ± 0.07	1.00 ± 0.00	0.77 ± 0.05	10
veterinary science	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	3

compared them to three classical baselines: a tag/term-overlap method (D4), supervised classifiers over column-name embeddings, and meta-feature-based models. In the few-shot setting, `gpt-4o` achieves accuracy and macro F1 on par with or slightly exceeding the best classical baseline, while DeepSeek-V3 also reaches competitive performance. At the same time, the embedding-based classifier maintains higher Top-3 recall, indicating complementarity between LLMs and traditional methods: LLMs provide sharper top-1 predictions, whereas classical models are stronger as broad-recall filters.

Ablation studies revealed that most of the information required for domain recognition is already present in the column headers: using only schema-level input yields few-shot performance that matches or slightly improves over configurations that include up to 50 data rows. Moreover, varying the number and sampling strategy of rows (5 versus 50, random versus diversity-based) showed that small, representative samples are sufficient, whereas longer tables mainly add verbosity and potential noise. Per-domain analyses further highlighted that LLMs perform best on domains with distinctive terminology and adequate support (e.g., medical, financial, environmental), but struggle on underrepresented or semantically overlapping domains (e.g., biology & genetics, synthetic systems, veterinary science), underscoring the sensitivity of LLM-based domain discovery to class imbalance and domain granularity.

Several directions emerge for future research. First, the benchmark could be expanded beyond OpenML to additional repositories and to multilingual settings, and the domain taxonomy could be refined into hierarchical or multi-label structures that better capture overlaps between areas. Second, hybrid systems that fuse LLM predictions with embedding-based or meta-feature signals may leverage the strengths of both, improving both top-1 accuracy and retrieval-oriented metrics such as Top-k recall. Third, automatic demonstration selection, prompt optimization, and robustness studies (e.g., under column renaming, missing values, or adversarial noise) may further stabilize LLM performance.

Finally, an important avenue is to integrate domain classification into end-to-end dataset search and recommendation pipelines, enabling domain-aware assistants that not only label tables but also drive model selection, data integration, and exploratory analysis over large, heterogeneous data lakes.

Acknowledgments

This research is financially supported by The Russian Scientific Foundation, Agreement № 24-71-10093, https://rscf.ru/en/project/24-71-10093/.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] E. Rahm, P. A. Bernstein, A survey of approaches to automatic schema matching, the VLDB Journal 10 (2001) 334–350.
[2] M. Hulsebos, K. Hu, M. Bakker, E. Zgraggen, A. Satyanarayan, T. Kraska, Ç. Demiralp, C. Hidalgo, Sherlock: A deep learning approach to semantic data type detection, in: Proceedings of the 25th ACM SIGKDD International Conference on knowledge discovery & data mining, 2019, pp. 1500–1508.
[3] M. Ota, H. Mueller, J. Freire, D. Srivastava, Data-driven domain discovery for structured datasets, Proceedings of the VLDB Endowment 13 (2020) 953–967.
[4] D. Zhang, Y. Suhara, J. Li, M. Hulsebos, Ç. Demiralp, W.-C. Tan, Sato: Contextual semantic type detection in tables, arXiv preprint arXiv:1911.06311 (2019).
[5] S. Langenecker, C. Sturm, C. Schalles, C. Binnig, Pythagoras: Semantic type detection of numerical data in enterprise data lakes., in: EDBT, 2024, pp. 725–733.
[6] W. Baazouzi, M. Kachroudi, S. Faiz, Kepler-asi: Semantic annotation for tabular data, 2024.
[7] P. Yin, G. Neubig, W.-t. Yih, S. Riedel, Tabert: Pretraining for joint understanding of textual and tabular data, arXiv preprint arXiv:2005.08314 (2020).
[8] J. Herzig, P. K. Nowak, T. Müller, F. Piccinno, J. M. Eisenschlos, Tapas: Weakly supervised table parsing via pre-training, arXiv preprint arXiv:2004.02349 (2020).
[9] Q. Liu, B. Chen, J. Guo, M. Ziyadi, Z. Lin, W. Chen, J.-G. Lou, Tapex: Table pre-training via learning a neural sql executor, arXiv preprint arXiv:2107.07653 (2021).
[10] Y. Sui, M. Zhou, M. Zhou, S. Han, 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.
[11] Z. Wang, H. Zhang, C.-L. Li, J. M. Eisenschlos, V. Perot, Z. Wang, L. Miculicich, Y. Fujii, J. Shang, C.-Y. Lee, et al., Chain-of-table: Evolving tables in the reasoning chain for table understanding, arXiv preprint arXiv:2401.04398 (2024).
[12] S. Han, W. Bruckner, Automatic demonstration selection for llm-based tabular data classification, arXiv preprint arXiv:2506.20451 (2025).
[13] H. Gong, Y. Sun, X. Feng, B. Qin, W. Bi, X. Liu, T. Liu, Tablegpt: Few-shot table-to-text generation with table structure reconstruction and content matching, in: Proceedings of the 28th International Conference on Computational Linguistics, 2020, pp. 1978–1988.
[14] X. Fang, W. Xu, F. A. Tan, J. Zhang, Z. Hu, Y. Qi, S. Nickleach, D. Socolinsky, S. Sengamedu, C. Faloutsos, Large language models (llms) on tabular data: Prediction, generation, and understanding–a survey, arXiv preprint arXiv:2402.17944 (2024).
[15] X. Zhang, D. Wang, L. Dou, Q. Zhu, W. Che, A survey of table reasoning with large language models, Frontiers of Computer Science 19 (2025) 199348.

[16] P. Li, S. Wang, H. Dai, Z. Chen, Z. Bao, B. D. Davison, A survey on open dataset search in the llm era: Retrospectives and perspectives, arXiv preprint arXiv:2509.00728 (2025).
[17] Sentence Transformers Team, all-mpnet-base-v2: A sentence transformers model for sentence embeddings, https://huggingface.co/sentence-transformers/all-mpnet-base-v2, 2023. Accessed: 2025-06-18.
[18] scikit-learn Developers, Tfidfvectorizer: Feature extraction from text in scikit-learn, https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html, 2025. Accessed: 2025-06-18.
[19] Gensim Developers, Word2vec: Word embeddings model in gensim, https://radimrehurek.com/gensim/models/word2vec.html, 2025. Accessed: 2025-06-18.
[20] scikit-learn Developers, Hdbscan: Hierarchical density-based spatial clustering in scikit-learn, https://scikit-learn.org/stable/modules/generated/sklearn.cluster.HDBSCAN.html, 2025. Accessed: 2025-06-18.
[21] scikit-learn Developers, Kmeans: K-means clustering in scikit-learn, https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html, 2025. Accessed: 2025-06-18.
[22] scikit-learn Developers, Agglomerativeclustering: Hierarchical clustering in scikit-learn, https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html, 2025. Accessed: 2025-06-18.
[23] scikit-learn Developers, Dbscan: Density-based spatial clustering in scikit-learn, https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html, 2025. Accessed: 2025-06-18.
[24] scikit-learn Developers, silhouette_score: Silhouette coefficient for cluster evaluation in scikit-learn, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html, 2025. Accessed: 2025-06-18.
[25] scikit-learn Developers, davies_bouldin_score: Davies-bouldin index for cluster evaluation in scikit-learn, https://scikit-learn.org/stable/modules/generated/sklearn.metrics.davies_bouldin_score.html, 2025. Accessed: 2025-06-18.
[26] M. Ota, H. Müller, J. Freire, D. Srivastava, Data-driven domain discovery for structured datasets, Proc. VLDB Endow. 13 (2020) 953–967. URL: https://doi.org/10.14778/3384345.3384346. doi:10.14778/3384345.3384346.
[27] E. Alcobaça, F. Siqueira, A. Rivolli, L. P. F. Garcia, J. T. Oliva, A. C. P. L. F. de Carvalho, Mfe: Towards reproducible meta-feature extraction, Journal of Machine Learning Research 21 (2020) 1–5. URL: http://jmlr.org/papers/v21/19-348.html.