TabEmbed: Benchmarking and Learning Generalist Embeddings for Tabular Understanding

Minjie Qiang[1,2]*, Mingming Zhang[2], Xiaoyi Bao[3], Xing Fu[2], Yu Cheng[2], Weiqiang Wang[2], Zhongqing Wang[1], Ningtao Wang[2]†

[1]Natural Language Processing Lab, Soochow University, Suzhou, China
[2]Ant Group, Hangzhou, China
[3]The Hong Kong Polytechnic University, Hong Kong, China

mjqiang@stu.suda.edu.cn, xiaoyi.bao@connect.polyu.hk, wangzq@suda.edu.cn
{mia.zmm, zicai.fx, cy122623, weiqiang.wwq, ningtao.nt}@antgroup.com

Abstract

Foundation models have established unified representations for natural language processing, yet this paradigm remains largely unexplored for tabular data. Existing methods face fundamental limitations: LLM-based approaches lack retrieval-compatible vector outputs, whereas text embedding models often fail to capture tabular structure and numerical semantics. To bridge this gap, we first introduce the Tabular Embedding Benchmark (TabBench), a comprehensive suite designed to evaluate the tabular understanding capability of embedding models. We then propose TabEmbed, the first generalist embedding model that unifies tabular classification and retrieval within a shared embedding space. By reformulating diverse tabular tasks as semantic matching problems, TabEmbed leverages large-scale contrastive learning with positive-aware hard negative mining to discern fine-grained structural and numerical nuances. Experimental results on TabBench demonstrate that TabEmbed significantly outperforms state-of-the-art text embedding models, establishing a new baseline for universal tabular representation learning. Code and datasets are publicly available at https://github.com/qiangminjie27/TabEmbed and https://huggingface.co/datasets/qiangminjie27/TabBench.

1 Introduction

Recently, foundation models have achieved remarkable success in establishing universal representations for Natural Language Processing (Wang et al., 2024; Yang et al., 2025), such as Retrieval-Augmented Generation (RAG) (Qiang et al., 2025), where dense text embeddings enable efficient semantic search through vector similarity computation. However, this unified representation paradigm has not been effectively adapted to tabular data. Existing research (Ye et al., 2025; Qu et al., 2025; Mueller et al., 2025) typically treats tabular classification and retrieval as distinct problems requiring specialized models. Consequently, the tabular domain lacks a shared embedding space capable of simultaneously addressing all tabular understanding tasks without task-specific architectures.

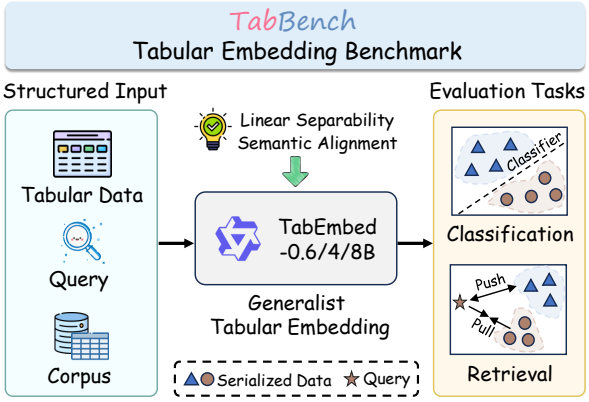

Figure 1: Overview of **TabBench** and **TabEmbed**.

Traditional tree-based models excel at tabular classification tasks but are constrained by fixed schemas, rendering them incompatible with zero-shot transfer and retrieval scenarios. Recent advances in large language models have shown considerable promise for tabular tasks (Gardner et al., 2024; Ye et al., 2025; Qu et al., 2025). However, these methods do not produce the dense, fixed-dimensional vectors required for vector databases and downstream retrieval applications. While general-purpose text embedding models (Zhang et al., 2025a; Yu et al., 2025; Zhang et al., 2025b) can generate such embeddings with remarkable success in text domains, they treat serialized tables as unstructured text, often failing to capture essential structural logic such as numerical magnitude and column-specific semantics. These constraints motivate the development of a generalist tabular em-

* Work done at Ant Group.
† Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 7355–7377
October 24-29, 2026 ©2026 Association for Computational Linguistics

bedding model that inherently understands tabular structure to handle various tabular understanding tasks within a shared embedding space.

However, training such a tabular embedding model presents three significant challenges. First, the absence of benchmarks specifically designed for tabular embeddings hinders systematic evaluation. Second, existing contrastive learning paradigms in the tabular domain are inadequate for unified understanding. Prior works typically rely on a row-to-row contrastive objective, where a data row serves as the anchor and is aligned with augmented views or other rows of the same class (e.g., SCARF (Bahri et al., 2021)). While this paradigm effectively separates classes, it forces the embedding space to collapse into coarse class clusters. By indiscriminately pulling together rows with divergent feature values simply because they share a target label, the model discards fine-grained structural semantics, logical constraints, and numerical magnitudes. Consequently, these representations fail to support precise semantic matching and retrieval; we verify this empirically in Appendix I.1. Finally, unifying classification and retrieval within a shared embedding space is non-trivial. Retrieval relies on semantic ranking to identify relevant data, whereas classification requires precise decision boundaries for label prediction.

The core value proposition of TabEmbed is to provide a universal, schema-agnostic representation that unifies diverse tabular tasks into a shared semantic space. This is an objective that traditional schema-bound models (e.g., XGBoost) cannot achieve without task-specific retraining. As shown in Figure 1, we first introduce TabBench, a comprehensive evaluation suite assessing numerical reasoning and retrieval capabilities. Then we propose TabEmbed, an embedding model that unifies classification and retrieval within a shared embedding space. To train this model, we depart from the suboptimal row-to-row paradigm and introduce a unified language-to-row contrastive framework. By synthesizing task-adaptive natural language queries as anchors, we reformulate diverse tasks into semantic matching problems. Enhanced by positive-aware hard negative mining, TabEmbed is compelled to discern fine-grained schema differences. Extensive experiments on TabBench demonstrate that TabEmbed significantly outperforms state-of-the-art text embeddings, establishing a new baseline for tabular understanding.

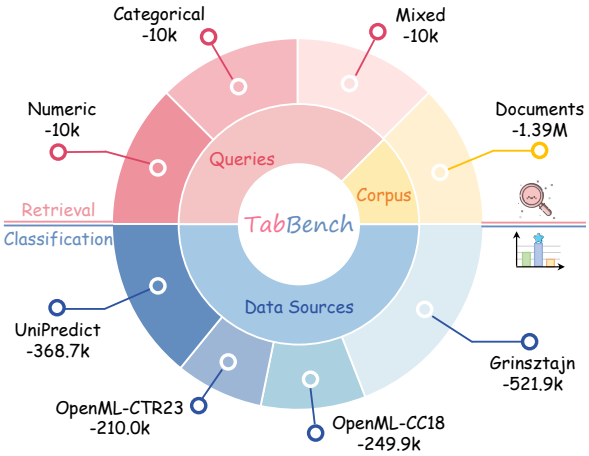

Figure 2: Data composition and statistics of TabBench.

2 The Tabular Embedding Benchmark

To rigorously evaluate the capabilities of embedding models in tabular understanding, we introduce the **Tabular Embedding Benchmark (TabBench)**. Building upon the high-quality data curation of the `tabula-8b-eval-suite` (Gardner et al., 2024), TabBench provides a comprehensive framework to assess two critical dimensions of tabular representation: linear separability (via classification) and semantic alignment (via retrieval). The benchmark aggregates diverse datasets from four authoritative repositories: **Grinsztajn** (Grinsztajn et al., 2022), **OpenML-CC18** (Bischl et al., 2017), **OpenML-CTR23** (Fischer et al., 2023), and **UniPredict** (Wang et al., 2023). The detailed composition of TabBench is illustrated in Figure 2. We implement a standardized pipeline for data serialization, task construction, and quality filtering.

2.1 Data Serialization

Bridging the modality gap between structured tabular data and large language models requires an effective serialization strategy (Hegselmann et al., 2023; Gardner et al., 2024). Formally, let a tabular row be represented as an ordered sequence of feature-value pairs $\mathbf{x} = ((h_1, v_1), \ldots, (h_C, v_C))$, where h_j denotes the column header and v_j is the corresponding cell value, with C being the total number of columns. We define a serialization function $\mathcal{S} : \mathcal{X} \rightarrow \mathcal{T}$ that maps $\mathbf{x}$ from the tabular space $\mathcal{X}$ to a natural language sequence in the text space $\mathcal{T}$ via string concatenation:

$$\mathcal{S}(\mathbf{x}) = \bigoplus_{j=1}^{C} (\text{“The } h_j \text{ is } \tilde{v}_j\text{.”}) , \qquad (1)$$

where $\oplus$ denotes the string concatenation operator, and $\tilde{v}_j$ represents the pre-processed string value. To maintain token efficiency and align with the context constraints of mainstream embedding models, we filter out rows that surpass the predefined maximum sequence length. Further details regarding the preprocessing and standardization of heterogeneous tabular data (e.g., numeric, temporal, and binary fields) are provided in Appendix B.4.

2.2 Evaluation Tasks

We formulate two distinct tasks to comprehensively evaluate the versatility of the learned embeddings within a shared vector space $\mathcal{Z} \in \mathbb{R}^d$. Let $f_\theta(\cdot)$ denote the embedding model that maps an input text sequence to a d-dimensional dense vector.

Tabular Classification This task evaluates the linear separability of the embeddings. We construct the evaluation suite by treating each source dataset as an independent classification task. Specifically, for a given tabular row, the input is the serialized text of its feature columns $\mathcal{S}(\mathbf{x}_i)$, and the output to predict is its corresponding discrete target label $y_i \in \mathcal{Y}$. Formally, given a dataset $\mathcal{D} = \{(\mathcal{S}(\mathbf{x}_i), y_i)\}_{i=1}^N$, we extract the frozen representations $\mathbf{z}_i = f_\theta(\mathcal{S}(\mathbf{x}_i))$. We then train an independent Logistic Regression classifier $g_\omega : \mathbb{R}^d \to \mathcal{Y}$ parameterized by ω for each dataset on top of the embeddings, optimized via:

$$\hat{\omega} = \arg\min_{\omega} \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{i \in \mathcal{D}_{\text{train}}} \mathcal{L}_{\text{CE}}(g_\omega(\mathbf{z}_i), y_i), \quad (2)$$

where $\mathcal{L}_{\text{CE}}$ denotes the cross-entropy loss. To ensure evaluation quality, we apply a strict filtering protocol: datasets are excluded if the label cardinality $|\mathcal{Y}| > 50$ or the label-to-sample ratio $|\mathcal{Y}|/N > 0.1$. For qualified datasets, we employ stratified sampling to partition data into training and testing splits, guaranteeing a minimum of two samples per class to mitigate cold-start issues for rare classes.

Tabular Retrieval Unlike classification, which assesses intra-dataset separability, the retrieval task evaluates the model's ability to align natural language queries with serialized rows across a heterogeneous global corpus $\mathcal{U}$. We construct $\mathcal{U}$ by aggregating rows from all datasets, capping each dataset's contribution at 10,000 samples to prevent distribution dominance.

To simulate realistic user intent, we propose a *seed-based query generation* pipeline. For a given "seed row" in the corpus, we generate a natural language query q following the template: *"Find records where c_1 and ... and c_k"*. This corresponds to a logical constraint condition $\Phi_q = c_1 \wedge c_2 \wedge \cdots \wedge c_k$, where each c_j represents an attribute constraint. The retrieval system ranks documents $d \in \mathcal{U}$ based on the cosine similarity score:

$$s(q, d) = \frac{f_\theta(q) \cdot f_\theta(d)}{\|f_\theta(q)\| \|f_\theta(d)\|}. \quad (3)$$

Let $\Phi_q(d) \in \{\text{True}, \text{False}\}$ denote whether document d satisfies the logical constraints in Φ_q. The goal is to retrieve the ideal target set $\mathcal{R}_q = \{d \in \mathcal{U} \mid \Phi_q(d) = \text{True}\}$. Based on the type of constraints, we define three query categories of increasing complexity:

- **Categorical Queries:** Assess exact-match semantics. Each constraint c_j enforces strict equality on discrete features (e.g., *"Status is Active"*).
- **Numeric Queries:** Test the understanding of magnitude and ranges. Each constraint c_j is generated by sampling a relational operator $\sim \in \{>, <, =\}$ and perturbing the original feature value (e.g., *"Price < 50.25"*).
- **Mixed Queries:** Evaluate complex reasoning by combining numeric and categorical constraints derived from the same row (e.g., *"Status is Active $\wedge$ Price < 50.25"*).

To ensure benchmark validity, we perform symbolic verification for every generated query, retaining only valid queries where the target set cardinality satisfies $|\mathcal{R}_q| \geq 5$. This process yields a balanced evaluation set, where each query contains 1 to 3 conditions (i.e., $k \in \{1, 2, 3\}$), covering diverse logical complexities.

3 TabEmbed: Unified Tabular Embedding Learning

To bridge the gap between structured data and semantic representation, we propose **TabEmbed**, a generalist embedding model trained within a unified framework that learns tabular representations by casting disparate downstream tasks into a shared contrastive paradigm. The overall framework is illustrated in Figure 3. Leveraging the massive scale of the T4 dataset (Gardner et al., 2024), we propose a novel language-to-row contrastive learning approach. Unlike conventional tabular methods that rely on row-to-row alignment, which often causes

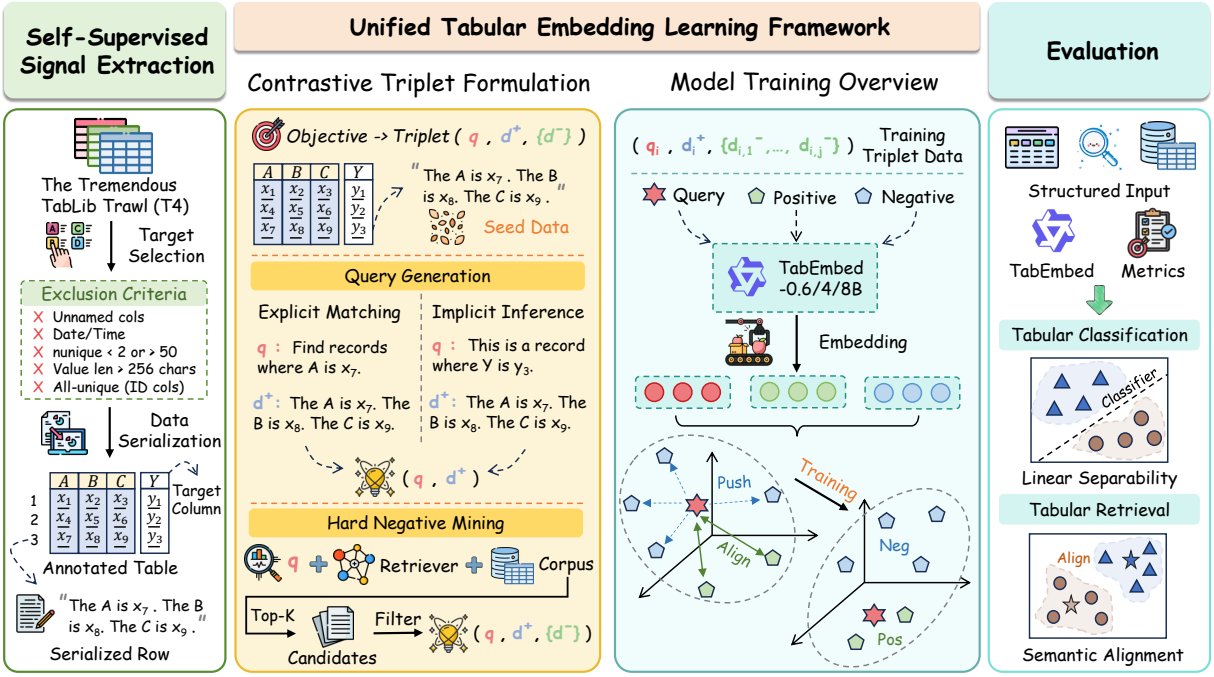

Figure 3: The overall framework of TabEmbed.

semantic collapse into coarse categories, our framework synthesizes natural language queries as anchors to construct diverse contrastive triplets. This strategy unifies disparate downstream capabilities into a shared semantic space while preserving fine-grained tabular structures.

3.1 Self-Supervised Signal Extraction

Since the T4 corpus lacks explicit task annotations, we employ an automated pipeline to transform raw tables into self-supervised training instances. We first dynamically identify a target column y within each table to serve as the prediction signal. To ensure the quality of these self-supervised signals, we apply a rigorous filtering protocol to exclude non-informative attributes (e.g., identifiers, timestamps) and prioritize targets with clear semantic boundaries. The Detailed pipeline is provided in Appendix H. To prevent information leakage and compel the model to learn latent dependencies, we apply a *target-masked serialization* strategy. Specifically, we strictly exclude the selected target y from the feature set and apply the serialization function $\mathcal{S}$ (defined in Section 2.1) to the remaining columns. This yields the serialized row $d = \mathcal{S}(\mathbf{x}_{-y})$, ensuring that the embedding captures the row's semantic content without revealing the ground truth label.

3.2 Contrastive Triplet Formulation

To overcome the limitations of traditional row-to-row instance discrimination, we formulate tabular representation learning as a language-to-row matching problem. Specifically, we optimize similarity within cross-modal triplets $(q, d^+, \{d^-\})$, where the anchor q is a dynamically generated natural language query expressing a specific tabular constraint or class intent, d^+ is the corresponding serialized row satisfying q, and $\{d^-\}$ are hard negatives. We construct these queries to cover both explicit signal matching and implicit semantic inference.

3.2.1 Task-Adaptive Query Generation

We generate synthetic queries q to model two complementary tasks using a shared data format:

Tabular Retrieval (Explicit Matching) The retrieval task aligns natural language constraints with rows that satisfy them. We leverage the query generation pipeline detailed in Section 2.2, which samples subsets of attributes from $\mathbf{x}_{-y}$ to form logical conditions spanning both numerical and categorical fields. Formally, for a serialized row d^+, we generate a query q_{ret} describing specific attribute constraints (e.g., *"Find records where Status is Active and Price less than 50.25"*). This forces the model to align natural language constraints with specific attribute values present in the input.

Embedding Models	#Params	Overall	Classification		Retrieval	
			Accuracy	F1	MRR@10	nDCG@10
Jina-Embeddings-v3	0.6B	41.48	60.33	46.11	32.49	26.98
Jasper-Token-Compression	0.6B	42.75	61.25	47.69	33.56	28.50
Qwen3-Embedding-0.6B	0.6B	44.92	62.81	50.32	36.00	30.56
TabEmbed-0.6B (Ours)	0.6B	**65.27**	**67.16**	**56.56**	**71.72**	**65.64**
F2LLM-4B	4B	48.02	64.92	52.48	40.60	34.08
Octen-Embedding-4B	4B	48.62	65.36	53.64	40.97	34.51
Qwen3-Embedding-4B	4B	48.91	65.09	52.72	42.04	35.76
TabEmbed-4B (Ours)	4B	**70.71**	**69.51**	**59.75**	**79.33**	**74.25**
SFR-Embedding-Mistral	7B	49.42	64.28	50.75	44.23	38.41
Linq-Embed-Mistral	7B	50.74	66.06	53.33	44.65	38.92
GTE-Qwen2-7B-Instruct	7B	51.27	64.67	51.76	47.44	41.19
Qwen3-Embedding-8B	8B	48.03	65.08	52.81	40.06	34.16
TabEmbed-8B (Ours)	8B	**71.62**	**69.88**	**60.19**	**80.58**	**75.83**

Table 1: **Tabular Embedding Benchmark (TabBench) Leaderboard.** We evaluate TabEmbed against state-of-the-art generalist text embedding models across three parameter scales. The best results are highlighted in **bold**.

Tabular Classification (Implicit Inference) The classification task aligns abstract label descriptions with rows that imply those labels. Unlike retrieval, the query content (the value of target y) is absent from the input d^+ and must be inferred solely from the correlations among the remaining features. For a hidden target column y with value v, we construct a descriptive label query q_{cls} (e.g., *"This is a record where y is v."*). This formulation encourages the model to cluster rows based on latent predictive features rather than surface-level token overlap.

3.2.2 Positive-Aware Hard Negative Mining

Simple in-batch negatives are insufficient for distinguishing numerically similar values or closely related classes. We implement an offline Hard Negative Mining strategy using a lightweight dense retriever (`Qwen3-Embedding-0.6B`). For every query q, we retrieve the Top-K candidates from the global corpus. Crucially, we employ a Positive-Aware Filtering mechanism: we strictly retain only those candidates that possess high semantic similarity to the query but explicitly violate the retrieval condition or belong to a different class label. These mined hard negatives d^- constitute the set of samples that are most easily confused with the positive d^+, ensuring the model learns sharp decision boundaries.

3.3 Training Objective

We optimize our model using the contrastive learning loss. Given a batch $\mathcal{B}$ containing B triplets $(q_i, d_i^+, \{d_{i,j}^-\}_{j=1}^H)$, where H is the number of mined hard negatives per query, the objective for query q_i is defined as:

$$\mathcal{L}_i = -\log \frac{e^{s(q_i, d_i^+)/\tau}}{e^{s(q_i, d_i^+)/\tau} + \sum_{d \in \mathcal{N}_i} e^{s(q_i, d)/\tau}}, \quad (4)$$

where $s(\cdot, \cdot)$ denotes the cosine similarity (as defined in Section 2.2), $\mathcal{N}_i$ includes both the H specific hard negatives and the in-batch negatives from other queries in $\mathcal{B}$, and τ is a temperature hyperparameter. This unified objective fosters a shared embedding space capable of generalizing across heterogeneous tabular understanding tasks.

4 Experiments

4.1 Implementation Details

We initialize TabEmbed using the Qwen3-Embedding family (Zhang et al., 2025b) across three scales: 0.6B, 4B, and 8B parameters. This selection allows us to evaluate the scalability of our unified training paradigm across varying computational regimes. The models are optimized using a contrastive learning objective within the Sentence-Transformers framework. We conduct evaluations on our proposed TabBench, with dataset statistics detailed in Figure 2. To construct the training data, we curate a balanced mixture of 500,000 retrieval and 100,000 classification contrastive triplets from the T4 dataset. For evaluation metrics, we report

Accuracy and **F1-Score** for the tabular prediction task, and **MRR@10** and **nDCG@10** for the tabular retrieval task. To provide a holistic measure of generalist capabilities, we also report an **Overall** score, computed as the macro-average of these four individual metrics. Further implementation details and evaluation protocols are provided in Appendix B.

4.2 Main Results

We evaluate TabEmbed on TabBench against a comprehensive suite of ten generalist text embedding models spanning three parameter scales (**0.6B**, **4B**, and **7B-8B**). Detailed specifications and citations for all baseline models are provided in Appendix I.

Table 1 presents the performance evaluation. The results demonstrate that TabEmbed achieves state-of-the-art performance across all parameter scales, significantly surpassing existing text embedding models. In **Tabular Retrieval**, TabEmbed yields substantial improvements, with the 0.6B model surpassing its Qwen3 backbone by over 35 points in MRR@10. This indicates that our unified contrastive learning paradigm effectively bridges the semantic gap between natural language queries and structured data, addressing a capability largely absent in text embeddings. In **Tabular Classification**, TabEmbed consistently improves accuracy and F1 scores, suggesting that the learned representations capture the fine-grained decision boundaries essential for linear separability. Crucially, our method exhibits remarkable parameter efficiency. TabEmbed-0.6B outperforms all baselines on the aggregate metric, including those in the 7B and 8B regimes. This finding suggests that domain-specific contrastive learning is more critical for tabular understanding than model scaling alone. Nevertheless, scaling TabEmbed from 0.6B to 8B yields consistent performance gains, confirming that our unified paradigm effectively leverages the capacity of larger foundation models to establish a new performance standard for tabular representation.

4.3 Performance on Diverse Backbones

To investigate the universality and robustness of our proposed training paradigm, we extend our evaluation beyond the Qwen3 family to a diverse set of backbone architectures. Specifically, we apply the unified contrastive learning paradigm to eight distinct foundation models, spanning different architectures (e.g., Qwen3, Mistral, and XLM-RoBERT) and parameter scales (ranging from 0.6B to 8B). We compare the performance of these models before and after applying our training framework, utilizing the original performance as baselines.

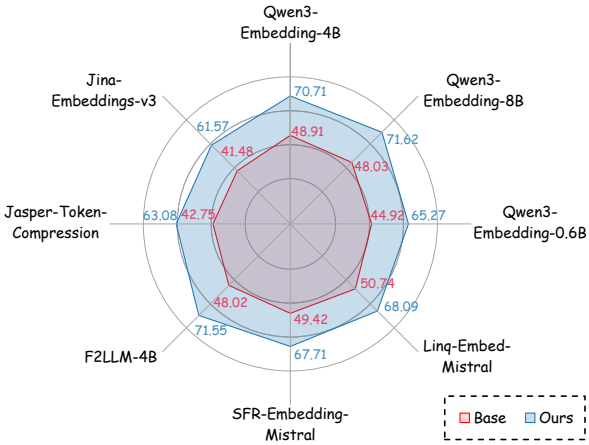

Figure 4: Performance comparison across backbone architectures using our proposed training paradigm. The performance metric is the Overall average score.

As illustrated in Figure 4, our approach consistently yields substantial performance improvements across all evaluated backbones, regardless of their architectural design or pre-training objective. Notably, models based on the Qwen3 architecture (e.g., F2LLM-4B) and the Mistral architecture (e.g., Linq-Embed-Mistral) exhibit significant enhancements, with Qwen3-Embedding-4B achieving the most significant improvement, surging from 48.91 to 70.71. Even for Jina-Embeddings-v3, which relies on an encoder-only XLM-RoBERT encoder architecture, our method achieves a remarkable gain of over 20 points (rising from 41.48 to 61.57). These results demonstrate that the improvements stem from the unified contrastive data paradigm rather than model-specific inductive biases, confirming that our paradigm effectively equips diverse text-based foundation models with generalized tabular understanding capabilities.

5 Analysis and Discussion

5.1 Fine-grained Analysis on Retrieval Capabilities

While the aggregate metrics demonstrate the overall superiority of TabEmbed, it is crucial to understand how the model behaves under different semantic modalities and logical complexities. To this end, we conduct a fine-grained breakdown of the retrieval performance on the Qwen3-Embedding-0.6B backbone, categorizing the test queries by type (Numeric, Categorical, and Mixed) and the number of logical constraints (from 1 to 3).

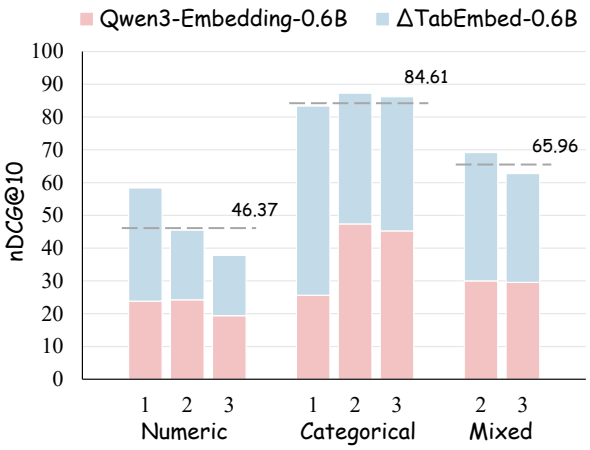

Figure 5: Fine-grained retrieval performance on TabBench (nDCG@10). The dashed lines indicate the average performance of TabEmbed for each query type.

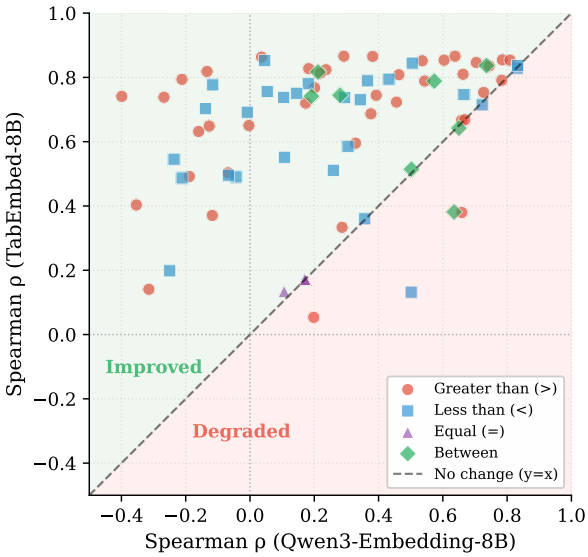

Figure 6: Pairwise comparison of numerical sensitivity between the baseline and TabEmbed. Each point represents a distinct test case, plotted by the Spearman correlation (ρ) between similarity scores and ground truth logic. Points in the green region indicate TabEmbed aligns significantly better with numerical constraints.

As illustrated in Figure 5, TabEmbed achieves consistent and substantial improvements across all query scenarios, yet the difficulty varies significantly by task type. The dashed lines representing the average performance reveal an inherent hierarchy of difficulty: *Categorical* queries are the most solvable (84.61), followed by *Mixed* (65.96), with *Numeric* queries presenting the greatest challenge (46.37). Crucially, the baseline model exhibits severe limitations in handling numerical queries, often failing to capture magnitude and range relationships. In contrast, TabEmbed contributes a massive performance gain in the Numeric category, effectively bridging the gap between text-based retrieval and numerical reasoning. Furthermore, regarding logical complexity, we observe that performance generally correlates with the number of constraints. For instance, in the Numeric setting, performance naturally decreases as the number of conditions increases from 1 to 3. Despite this increased difficulty, TabEmbed maintains robust performance, validating its ability to handle complex, multi-condition logical intersections within the embedding space.

5.2 Numerical Sensitivity Analysis

Standard text embedding models often treat numbers as arbitrary tokens, lacking awareness of magnitude and inequality. To investigate whether TabEmbed has acquired genuine numerical reasoning capabilities beyond surface-level token matching, we conduct a **Numerical Sensitivity Test**. Specifically, for a given query containing a numerical constraint (e.g., $q =$ *"Revenue greater than 500"*), we generate a sequence of candidate values x ranging from small to large. We then compute the Spearman correlation (ρ) between the cosine similarity $\text{sim}(q, x)$ and the ground truth logical satisfaction (i.e., the ideal curve should step up when $x > 500$).

Figure 6 visualizes the pairwise comparison of these correlation coefficients across diverse test cases, including inequalities ($>, <$), equality ($=$), and range queries (*Between*). The results reveal a distinct performance gap: the baseline Qwen3-Embedding (X-axis) frequently exhibits near-zero or weakly positive correlations, suggesting it struggles to distinguish between numerically valid and invalid candidates. In contrast, TabEmbed (Y-axis) shifts the majority of test cases into the upper-left "Improved" region, with many cases achieving high correlations ($\rho > 0.8$). This substantial shift indicates that our model has successfully internalized numerical semantics, mapping mathematically close or logically valid values to closer proximity in the vector space. Detailed visualizations of similarity curves are provided in Appendix E.

5.3 Visualization of Embedding Spaces

To provide a qualitative assessment of the learned representations, we project the high-dimensional embeddings into a 2D space using PCA and t-SNE. We visualize the geometric structures for both classification and retrieval tasks, comparing the baseline Qwen3-Embedding-8B against TabEmbed-8B.

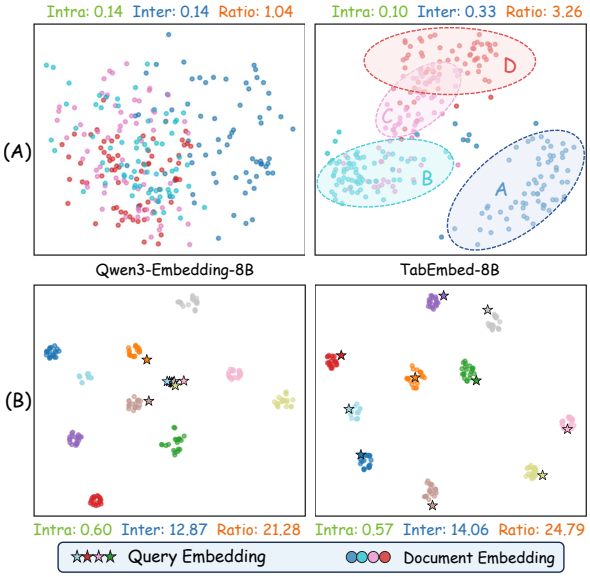

Figure 7: Visualization comparing Qwen3-Embedding-8B (left) and TabEmbed-8B (right) on tabular classification (A) and tabular retrieval (B) tasks. We report the Cluster Ratio to quantify the clustering quality.

To quantify the clustering quality, we report the *Cluster Ratio*, defined as the ratio of inter-cluster distance to intra-cluster distance, where a higher ratio indicates better separability.

Figure 7(A) illustrates the feature space for classification. The baseline exhibits a highly entangled distribution (Ratio: 1.04) with significant overlap between classes, suggesting a failure to capture discriminative boundaries. In contrast, TabEmbed effectively disentangles these classes into well-separated clusters (regions A-D), substantially increasing the Cluster Ratio to 3.26. This confirms that our contrastive paradigm imparts linear separability to the embedding space, enabling efficient downstream classification.

Figure 7(B) visualizes semantic alignment for retrieval. Although the baseline exhibits partial alignment capabilities, many queries ($\star$) remain drifting away from their target document clusters. In contrast, TabEmbed consistently anchors queries within their corresponding groups and pulls relevant documents tighter around query centers, producing significantly more compact clusters (Intra: 0.60 $\rightarrow$ 0.57) and higher separability (Ratio: 21.28 $\rightarrow$ 24.79). This demonstrates that TabEmbed learns a precise alignment between natural language constraints and structured tabular data, effectively correcting the misalignment observed in the baseline.

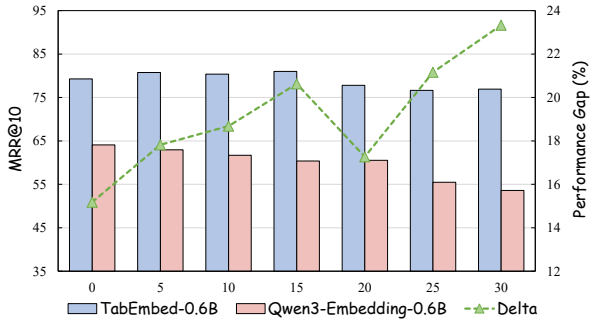

Figure 8: Robustness analysis against irrelevant table columns. We incrementally inject noise columns (0 to 30) into the documents while maintaining fixed queries.

5.4 Robustness to Irrelevant Table Columns

Real-world tabular data is often characterized by high dimensionality, where a user's query typically targets only a small subset of columns (e.g., filtering by *Price* and *City*) while ignoring numerous irrelevant attributes. Standard text embedding models are susceptible to **semantic dilution**, where irrelevant text diminishes the weight of target information in high-dimensional tables. To evaluate robustness against such structural noise, we incrementally inject up to 30 irrelevant columns into documents initially containing 15 columns and observe the degradation in MRR@10.

As shown in Figure 8, the baseline Qwen3-Embedding exhibits a marked sensitivity to noise. Its performance declines steadily as the number of irrelevant columns increases, dropping from $\sim$64% to below 55%. This confirms that without structural awareness, the model struggles to attend to the relevant signal amidst a growing volume of noise tokens. In contrast, TabEmbed demonstrates exceptional stability, consistently maintaining an MRR@10 above 75% even when 30 irrelevant columns are added. Crucially, the green dashed line highlights that the performance gap (Δ) between the two models widens monotonically from $\sim$15% at the noise-free baseline to over 23% at the maximum noise level. This result suggests that TabEmbed has effectively learned an implicit structural attention mechanism, enabling it to selectively align query constraints with matching columns while filtering out unrelated tabular context.

6 Conclusion

We introduced **TabEmbed**, a unified embedding model that bridges the gap between tabular classification and retrieval. Supported by our proposed **TabBench** benchmark, we demonstrated that stan-

dard text embeddings struggle with tabular structure and numerical semantics. TabEmbed addresses these challenges through a unified contrastive learning paradigm, utilizing task-adaptive query generation and hard negative mining to learn discriminative representations. Our experiments reveal that TabEmbed achieves state-of-the-art performance, with the 0.6B model surpassing significantly larger baselines. This work establishes a baseline for generalist tabular embeddings, demonstrating that rigorous domain alignment is a more effective path to tabular intelligence than parameter scaling alone.

Limitations

Despite the promising results of TabEmbed on the proposed benchmark, there are several limitations to our current study. First, due to the substantial scale of TabBench (comprising over 300 datasets) and budgetary constraints, we did not include commercial closed-source embedding APIs (e.g., Google `Gemini Embedding` (Lee et al., 2025)) in our evaluation. While our comparison covers a wide range of state-of-the-art open-source models, a comprehensive benchmarking against these commercial systems remains a direction for future research. Second, our method relies on serializing tabular data into natural language sequences. For extremely wide tables with hundreds of columns, the serialized text may exceed the maximum context window of the backbone models, potentially leading to information truncation. Developing more token-efficient serialization strategies or employing long-context architectures to handle ultra-wide tables is an avenue we plan to explore in future work.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No. 62376178), Jiangsu Key Laboratory of Language Computing (JSLCKeyLab 202500003), and the Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions. This work was also supported by Ant Group Research Intern Program.

References

Sercan Ö Arik and Tomas Pfister. 2021. TabNet: Attentive interpretable tabular learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 6679–6687.

Dara Bahri, Heinrich Jiang, Yi Tay, and Donald Metzler. 2021. SCARF: Self-supervised contrastive learning using random feature corruption. *arXiv preprint arXiv:2106.15147*.

Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Pieter Gijsbers, Frank Hutter, Michel Lang, Rafael G Mantovani, Jan N van Rijn, and Joaquin Vanschoren. 2017. OpenML benchmarking suites. *arXiv preprint arXiv:1708.03731*.

Wenhu Chen, Ming-Wei Chang, Eva Schlinger, William Wang, and William W Cohen. 2020. Open question answering over tables and text. *arXiv preprint arXiv:2010.10439*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Gus Eggert, Kevin Huo, Mike Biven, and Justin Waugh. 2023. TabLib: A dataset of 627M tables with context. *arXiv preprint arXiv:2310.07875*.

Sebastian Felix Fischer, Matthias Feurer, and Bernd Bischl. 2023. OpenML-CTR23: A curated tabular regression benchmarking suite. In *AutoML Conference 2023 (Workshop)*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Josh Gardner, Juan C Perdomo, and Ludwig Schmidt. 2024. Large scale transfer learning for tabular data via language modeling. *Advances in Neural Information Processing Systems*, 37:45155–45205.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. 2021. Revisiting deep learning models for tabular data. *Advances in neural information processing systems*, 34:18932–18943.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. 2022. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in neural information processing systems*, 35:507–520.

Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. 2022. Accelerate: Training and inference at scale made simple, efficient and adaptable. https://github.com/huggingface/accelerate.

Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. 2023. TabLLM: Few-shot classification of tabular data with large language models. In *International conference on artificial intelligence and statistics*, pages 5549–5581. PMLR.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 4320–4333.

Dhiraj Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka, Jianyu Huang, Hector Yuen, and 1 others. 2019. A study of BFLOAT16 for deep learning training. *arXiv preprint arXiv:1905.12322*.

Junseong Kim, Seolhwa Lee, Jihoon Kwon, Sangmo Gu, Yejin Kim, Minkyung Cho, Jy-yong Sohn, and Chanyeol Choi. 2024. Linq-Embed-Mistral: Elevating text retrieval with improved GPT data through task-specific control and quality refinement. Linq AI Research Blog.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. NV-Embed: Improved techniques for training LLMs as generalist embedding models. *arXiv preprint arXiv:2405.17428*.

Jinhyuk Lee, Feiyang Chen, Sahil Dua, Daniel Cer, Madhuri Shanbhogue, Iftekhar Naim, Gustavo Hernández Ábrego, Zhe Li, Kaifeng Chen, Henrique Schechter Vera, and 1 others. 2025. Gemini embedding: Generalizable embeddings from Gemini. *arXiv preprint arXiv:2503.07891*.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*.

Duncan McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C, Ganesh Ramakrishnan, Micah Goldblum, and Colin White. 2023. When do neural nets outperform boosted trees on tabular data? *Advances in Neural Information Processing Systems*, 36:76336–76369.

Rui Meng, Ye Liu, Shafiq Rayhan Joty, Caiming Xiong, Yingbo Zhou, and Semih Yavuz. 2024. SFR-Embedding-Mistral: Enhance text retrieval with transfer learning. Salesforce AI Research Blog.

Andreas C Mueller, Carlo A Curino, and Raghu Ramakrishnan. 2025. MotherNet: Fast training and inference via hyper-network transformers. In *The Thirteenth International Conference on Learning Representations*.

Niklas Muennighoff. 2022. SGPT: GPT sentence embeddings for semantic search. *arXiv preprint arXiv:2202.08904*.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2023. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480.

Minjie Qiang, Zhongqing Wang, Shoushan Li, and Guodong Zhou. 2025. Exploring unified training framework for multimodal user profiling. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 1699–1710.

Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. 2025. TabICL: A tabular foundation model for in-context learning on large data. In *ICML 2025-Forty-Second International Conference on Machine Learning*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. ZeRO: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese BERT-networks. *arXiv preprint arXiv:1908.10084*.

Saba Sturua, Isabelle Mohr, Mohammad Kalim Akram, Michael Günther, Bo Wang, Markus Krimmel, Feng Wang, Georgios Mastrapas, Andreas Koukounas, Nan Wang, and 1 others. 2024. jina-embeddings-v3: Multilingual embeddings with task LoRA. *arXiv preprint arXiv:2409.10173*.

Octen Team. 2025. Octen series: Optimizing embedding models to #1 on RTEB leaderboard.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11897–11916.

Ruiyu Wang, Zifeng Wang, and Jimeng Sun. 2023. UniPredict: Large language models are universal tabular classifiers. *arXiv preprint arXiv:2310.03266*.

Zifeng Wang and Jimeng Sun. 2022. TransTab: Learning transferable tabular transformers across tables. *Advances in Neural Information Processing Systems*, 35:2902–2915.

Xumeng Wen, Han Zhang, Shun Zheng, Wei Xu, and Jiang Bian. 2024. From supervised to generative: A novel paradigm for tabular deep learning with large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3323–3333.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-Pack: Packed resources for general Chinese embeddings. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pages 641–649.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Han-Jia Ye, Si-Yang Liu, and Wei-Lun Chao. 2025. A closer look at TabPFN v2: Understanding its strengths and extending its capabilities. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 8413–8426.

Peng Yu, En Xu, Bin Chen, Haibiao Chen, and Yinfei Xu. 2025. QZhou-Embedding technical report. *Preprint*, arXiv:2508.21632.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, and 1 others. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 3911–3921.

Dun Zhang, Ziyang Zeng, Yudong Zhou, and Shuyang Lu. 2025a. Jasper-Token-Compression-600M technical report. *Preprint*, arXiv:2511.14405.

Tianping Zhang, Shaowen Wang, Shuicheng Yan, Jian Li, and Qian Liu. 2023. Generative table pre-training empowers models for tabular prediction. *arXiv preprint arXiv:2305.09696*.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025b. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

Ziyin Zhang, Zihan Liao, Hang Yu, Peng Di, and Rui Wang. 2025c. F2LLM technical report: Matching SOTA embedding performance with 6 million open-source data. *arXiv preprint arXiv:2510.02294*.

Bingzhao Zhu, Xingjian Shi, Nick Erickson, Mu Li, George Karypis, and Mahsa Shoaran. 2023. XTab: Cross-table pretraining for tabular transformers. *arXiv preprint arXiv:2305.06090*.

A Related Work

A.1 Embedding Models and Tabular Representation

Text embedding research has evolved from encoder-based architectures to Large Language Models (LLMs). Early works adapted BERT (Devlin et al., 2019) and T5 (Raffel et al., 2020) via contrastive learning (Reimers and Gurevych, 2019; Gao et al., 2021), with recent models like E5 (Wang et al., 2022), BGE (Xiao et al., 2024), and GTE (Li et al., 2023) achieving scalability through multi-stage training. However, limited capacity for complex schemas prompted a shift toward decoder-only LLMs. Approaches like SGPT (Muennighoff, 2022) and E5-Mistral (Wang et al., 2024) demonstrated the efficacy of generative backbones, while subsequent innovations (Lee et al., 2024; Zhang et al., 2025b) further optimized bidirectional information flow. Despite these advancements, existing generalist text models typically process serialized tables as unstructured text, lacking the specific optimization for structural reasoning and numerical understanding required for precise tabular retrieval.

Parallel to text embeddings, representation learning specifically designed for tabular data has also seen significant progress. Early deep tabular models, such as FT-Transformer (Gorishniy et al., 2021) and TabNet (Arik and Pfister, 2021), introduced specialized attention mechanisms and feature tokenization to handle heterogeneous columns. To achieve transferability across different tables, models like TransTab (Wang and Sun, 2022) and XTab (Zhu et al., 2023) proposed shared tokenizers or cross-table pre-training strategies. More recently, LLMs have been directly applied to tabular tasks. Approaches such as TabLLM (Hegselmann et al., 2023), UniPredict (Wang et al., 2023), and TaP-TaP (Zhang et al., 2023) serialize tabular rows into

natural language prompts to perform zero-shot or few-shot classification via text generation. However, these methods are primarily confined to generative or predictive paradigms. They either require task-specific architectures or rely on autoregressive decoding for label prediction, failing to produce the dense, fixed-dimensional vectors necessary for efficient semantic search and retrieval in vector databases.

A.2 Benchmarks for Text and Tabular Tasks

Current evaluation protocols remain bifurcated between unstructured text and supervised tabular classification. In the text domain, standards like BEIR (Thakur et al., 2021) and MTEB (Muennighoff et al., 2023) have driven the progress of retrieval and embedding models, yet they lack dedicated structured data scenarios involving numerical and categorical constraints. Conversely, tabular machine learning benchmarks, such as OpenML-CC18 (Bischl et al., 2017) and the Grinsztajn suite (Grinsztajn et al., 2022), focus primarily on comparing decision trees against neural networks on fixed intra-dataset classification splits.

Another related line of evaluation includes Table Question Answering (Table QA) and Semantic Parsing benchmarks, such as WikiTableQuestions (Pasupat and Liang, 2015), Spider (Yu et al., 2018), and OTT-QA (Chen et al., 2020). Pre-trained models like TAPAS (Herzig et al., 2020) and TaBERT (Yin et al., 2020) were evaluated on these datasets to measure their joint understanding of text and tables. However, these benchmarks are heavily biased toward answering natural language questions over Wikipedia-style, text-heavy tables or translating text to SQL queries. They do not systematically evaluate a model's ability to map raw, heterogeneous tabular rows into a general-purpose embedding space.

While recent massive corpora (Eggert et al., 2023; Gardner et al., 2024) have enabled large-scale transfer learning suites like TabZilla (McElfresh et al., 2023) and GTL (Wen et al., 2024), these evaluations still treat tabular data learning strictly as an isolated prediction problem. To bridge this critical gap, we introduce TabBench, a comprehensive evaluation suite that simultaneously assesses numerical reasoning, semantic alignment, and linear separability, and propose TabEmbed, a generalist model that unifies these diverse tabular tasks within a single shared vector space.

Model	Layers	Hidden Dim	Max Context
TabEmbed-0.6B	28	1024	32K
TabEmbed-4B	36	2560	32K
TabEmbed-8B	36	4096	32K

Table 2: Architectural specifications of the TabEmbed model family. All variants are initialized from the corresponding Qwen3-Embedding checkpoints and inherit their structural configurations.

B Detailed Implementation

B.1 Training Configurations

Our training pipeline is built upon the HuggingFace `Accelerate` library (Gugger et al., 2022) and the `Sentence-Transformers` framework (Reimers and Gurevych, 2019). We fine-tune all backbone models for 2 epochs using the contrastive Multiple Negatives Ranking Loss (MNRL) with a temperature parameter $\tau = 0.05$. The training objective is to maximize the cosine similarity between the query q and the positive document d^+, while minimizing the similarity with both in-batch negatives and the mined hard negatives described in Section 3.2.

We utilize the AdamW optimizer with a learning rate of 1×10^{-5} and a linear learning rate decay schedule, following a warmup period spanning the first 10% of the total training steps. To balance computational efficiency with the need to capture long-range tabular dependencies, we set the maximum sequence length to 1024 tokens. The global batch size is set to 256. For the dataset composition, we sample 500,000 retrieval triplets and 100,000 classification triplets from the processed T4 corpus. To ensure training stability, particularly for the 8B parameter models, we employ BFloat16 (BF16) mixed-precision training (Kalamkar et al., 2019).

B.2 Model Architecture

TabEmbed is built upon the dense decoder-only architecture of the Qwen3-Embedding family. We release TabEmbed in three sizes (0.6B, 4B, and 8B) to cater to diverse computational constraints. While our fine-tuning protocol utilizes a context length of 1,024 tokens to optimize training throughput, the underlying architecture supports distinctively long contexts (up to 32K tokens) and variable embedding dimensions. The detailed architectural specifications for each model variant are summarized in Table 2.

Category	Count	# Samples / Corpus
CLASSIFICATION BENCHMARKS		
Grinsztajn	56	521,889
OpenML-CC18	66	249,939
OpenML-CTR23	34	210,026
UniPredict	155	386,618
Classification Total	*311*	*1,368,472*
RETRIEVAL BENCHMARKS		
Corpus	—	1,394,247
Numeric Queries	10,000	—
Categorical Queries	10,000	—
Mixed Queries	10,000	—
Retrieval Total	*30,000*	*1,394,247*

Table 3: Statistics of the Tabular Embedding Benchmark (TabBench). The benchmark aggregates datasets from four diverse high-quality sources for classification and constructs a large-scale corpus for retrieval tasks.

B.3 Evaluation Protocols

To ensure rigorous and reproducible evaluation, we fix the random seed to 42 across all experiments. Table 3 summarizes the scale and composition of our evaluation benchmarks (TabBench). The specific evaluation protocols for the two tasks are as follows:

Tabular Classification (Linear Probing) We assess the linear separability of the embedding space by training a lightweight classifier on top of fixed representations. Specifically, for each evaluated embedding model, we extract frozen dense vectors for all samples. Then, for each individual dataset, we train an independent Logistic Regression classifier using the `scikit-learn` library solely on that dataset's training split embeddings, and evaluate its performance on the corresponding test split. It is important to note that there is no shared classifier across datasets; every embedding model is equipped with its own optimized probe per dataset to fairly evaluate its representation quality. The classifiers are consistently configured with a maximum of 1,000 iterations (`max_iter=1000`) and a fixed random seed (`random_state=42`) to ensure convergence and strict reproducibility. We report **Accuracy** and **Macro-F1 Score** to account for potential class imbalances in the source datasets.

Tabular Retrieval (Dense Retrieval) For the retrieval task, we utilize the `Faiss` library for efficient vector similarity search. We employ an exact search strategy using the `IndexFlatIP` index (Inner Product), which corresponds to Cosine Similarity as all embeddings are L_2-normalized prior to indexing. For each query, we retrieve the top-k most similar documents from the corpus. Performance is measured using **MRR@10** (Mean Reciprocal Rank) and **nDCG@10** (Normalized Discounted Cumulative Gain), which evaluate the ranking quality of the relevant ground-truth rows. While our main results report metrics at $k = 10$, we also compute Recall and Precision at various cutoffs ($k \in \{1, 5, 10, 20, 50, 100\}$) for comprehensive analysis.

Algorithm 1 Heterogeneous Tabular Data Serialization

Input: A tabular row $\mathbf{x} = \{(h_1, v_1), (h_2, v_2), \ldots, (h_C, v_C)\}$, numerical precision π (e.g., $\pi = 2$).
Output: Serialized natural language sequence $\mathcal{S}(\mathbf{x})$.

Initialize an empty sequence list $\mathcal{L} \leftarrow \emptyset$.
for each feature-value pair $(h_j, v_j) \in \mathbf{x}$ **do**
  if v_j is Null **or** NaN **then**
    $\tilde{v}_j \leftarrow$ "unknown"
  else if v_j is Numeric **then**
    $\tilde{v}_j \leftarrow \text{Round}(v_j, \pi)$
    if $\tilde{v}_j$ has no fractional part **then**
      $\tilde{v}_j \leftarrow \text{Integer}(\tilde{v}_j)$
    end if
  else if v_j is Date **or** Timestamp **then**
    $\tilde{v}_j \leftarrow \text{ISO8601Format}(v_j)$
  else if v_j is Binary (Bytes) **then**
    $\tilde{v}_j \leftarrow \text{DecodeUTF8}(v_j)$ with fallback to Latin-1
  else
    $\tilde{v}_j \leftarrow \text{String}(v_j)$
    Strip leading/trailing whitespaces and trailing periods from $\tilde{v}_j$.
  end if
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{\text{"The } h_j \text{ is } \tilde{v}_j\text{."}\}$
end for
$\mathcal{S}(\mathbf{x}) \leftarrow \text{Join}(\mathcal{L}, \text{delimiter} = \text{" "})$
return $\mathcal{S}(\mathbf{x})$

B.4 Heterogeneous Data Serialization Details

Real-world tabular datasets often comprise a wide variety of data types, which traditional schema-bound models (such as tree-based models) handle via specialized encoding layers. To handle such heterogeneous tabular fields within a single gener-

alist architecture, our serialization pipeline unifies all data types into a standardized natural language format prior to string concatenation. The detailed step-by-step procedure is outlined in Algorithm 1.

Specifically, we apply the following preprocessing rules to obtain the string representation $\tilde{v}_j$ for each cell value v_j:

- **Numeric Data:** Continuous and discrete numerical values (including binary 0/1 integers) are rounded to a specified precision (default two decimal places) to preserve magnitude without excessive tokenization overhead. Integers are preserved without trailing decimals.
- **Categorical and Textual Data:** These fields are converted directly to their literal string representations, stripping trailing periods to avoid punctuation conflicts in the serialization template.
- **Temporal Data:** Dates and timestamps are standardized into the ISO 8601 format (e.g., *"2025-03-01T00:00:00"*) to provide a consistent temporal syntax.
- **Binary Data:** Raw binary streams, where applicable, are decoded into text via UTF-8 or Latin-1 encodings.

By converting these diverse fields into a unified natural language context, TabEmbed leverages the pre-trained semantic knowledge of the LLM backbone to comprehend heterogeneous data simultaneously, effectively eliminating the need for modality-specific feature engineering.

B.5 Hardware and Infrastructure

All experiments are conducted on a high-performance computing cluster equipped with 16 PPU-810E accelerators, each possessing 96GB of high-bandwidth memory. To efficiently fine-tune the large-scale models (up to 8 billion parameters), we implement a composite optimization strategy. This includes **DeepSpeed ZeRO Stage 2** (Rajbhandari et al., 2020) for optimizer state partitioning and **Gradient Checkpointing** to reduce memory fragmentation. The multi-GPU training is orchestrated via **Distributed Data Parallelism**, ensuring linear scaling of the effective batch size.

B.6 Sensitivity to the Negative Mining Model

Our hard negatives are proposed by a dense retriever (`Qwen3-Embedding-0.6B`) and subsequently filtered by symbolic constraint checking, which raises the question of whether the mined negatives encode a preference for the Qwen family. To test this directly, we re-mined the entire negative pool with two encoders drawn from unrelated families, BGE-large-en-v1.5 (Xiao et al., 2024) and E5-Mistral-7B (Wang et al., 2024), and retrained the 0.6B model from the same backbone under identical hyperparameters.

Mining Model	Overall	Δ
Qwen3-Embedding-0.6B (released)	65.27	—
BGE-large-en-v1.5	65.75	+0.48
E5-Mistral-7B	65.51	+0.24

Table 4: Effect of replacing the negative mining model. All variants are trained from the same backbone with identical hyperparameters and differ only in the retriever used to propose hard negative candidates.

As shown in Table 4, the three variants fall within 0.48 Overall of one another, and the two non-Qwen miners in fact score marginally higher than the released configuration. This is consistent with the design of the mining stage: the retriever only *proposes* candidates, whereas acceptance is decided symbolically, so the retained negatives carry no embedding-family preference. We note one practical artifact of this substitution. Stronger miners produce sharper similarity distributions among same-label rows, so fewer classification triplets survive the score filter, yielding 254K, 79.6K, and 19.1K triplets for Qwen3, BGE, and E5-Mistral respectively. For E5-Mistral this imbalance destabilizes the joint early-stopping criterion, which averages the retrieval and classification losses; we therefore report the checkpoint at which the retrieval objective converges.

B.7 Per-Source Breakdown and Leave-One-Source-Out Stability

TabBench aggregates datasets from four repositories with markedly different characteristics, so an aggregate score could in principle be dominated by a single source. Table 5 reports classification accuracy separately for each source, comparing TabEmbed-8B against its Qwen3-Embedding-8B backbone.

TabEmbed-8B improves over its backbone on all four sources, with the largest gain of 6.6 points on the numerically dense OpenML-CTR23 suite and the smallest of 4.3 points on Grinsztajn. To verify that no single source drives the aggregate result, Table 6 recomputes the Overall score with

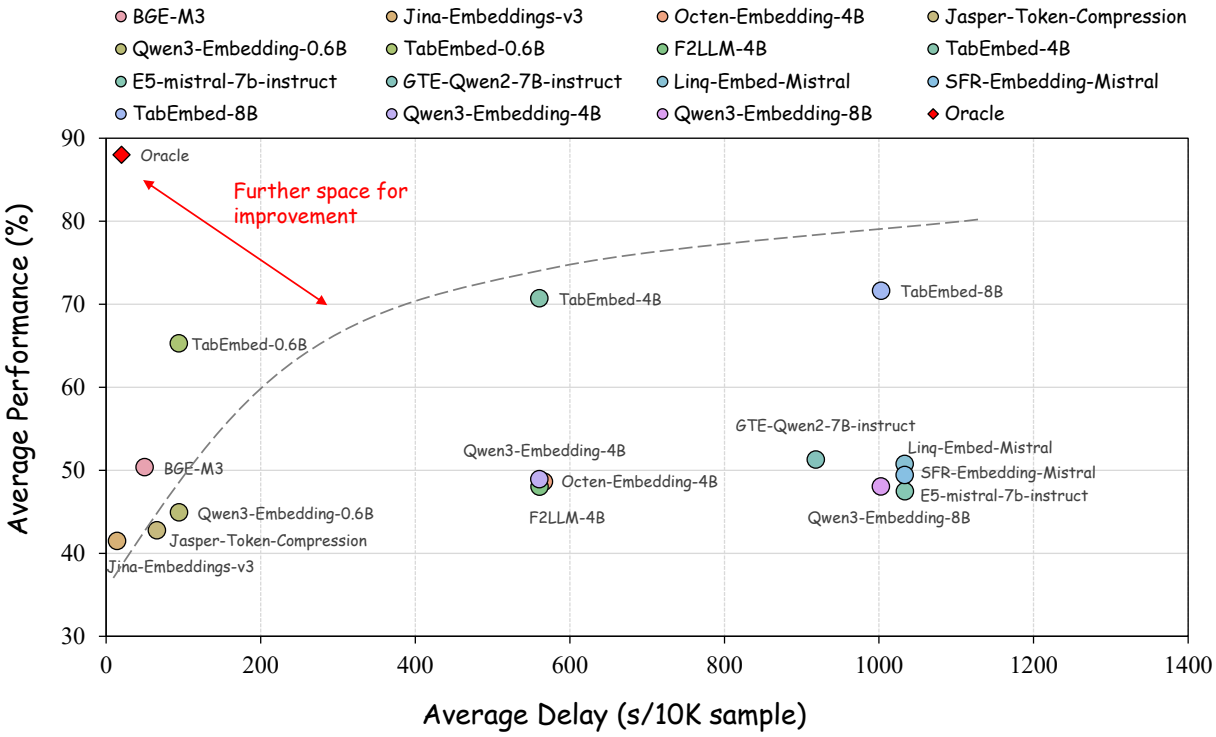

Figure 9: Performance vs. Latency trade-off. The Y-axis represents the overall average performance (computed as the macro-average of the four evaluation metrics) on TabBench, while the X-axis denotes the average inference delay (seconds per 10,000 samples).

Source	#Sets	Qwen3-8B	TabEmbed-8B
Grinsztajn	56	62.8	67.1
OpenML-CC18	66	63.1	67.5
OpenML-CTR23	34	53.7	60.3
UniPredict	155	69.2	74.0
Weighted average	**311**	**65.1**	**69.9**

Table 5: Classification accuracy broken down by source repository. The weighted average reproduces the aggregate accuracy reported in Table 1.

Aggregation	Qwen3-8B	TabEmbed-8B	Gap
Full benchmark	48.03	71.62	+23.59
– Grinsztajn	48.07	72.79	+24.72
– OpenML-CC18	48.58	73.35	+24.77
– OpenML-CTR23	47.55	71.81	+24.26
– UniPredict	47.50	66.49	+18.99
Maximum deviation	0.55	5.13	—

Table 6: Leave-one-source-out re-aggregation of the Overall score. Each row removes one source repository and recomputes the metric over the remainder.

each source removed in turn.

The ordering never changes: TabEmbed-8B leads its backbone by between 18.99 and 24.77 points under every re-aggregation. Removing any of the three smaller sources moves the Overall score by at most 1.7 points. Removing UniPredict shifts it by 5.13, which is expected given that UniPredict contributes 155 of the 311 datasets and is the source on which TabEmbed performs best (74.0 accuracy); the effect reflects sample composition rather than a narrow dependency, as the margin over the backbone remains close to 19 points even in that setting.

C Train-Test Overlap and Contamination Audit

TabEmbed is trained on tables drawn from the T4 corpus (Gardner et al., 2024), a large web-crawled collection, whereas TabBench aggregates datasets from curated public repositories. Because the two pools are constructed independently, overlap is possible in principle, and we audit it explicitly rather than assume its absence.

We reconstruct the exact set of 1,750 training tables and compare them against all 311 benchmark datasets at increasing levels of granularity. Following the schema protocol introduced with the T4 corpus (Gardner et al., 2024), a *fuzzy* match holds when every benchmark column name is contained in a training table's column set (case-insensitive), and a *strict* match additionally requires the two col-

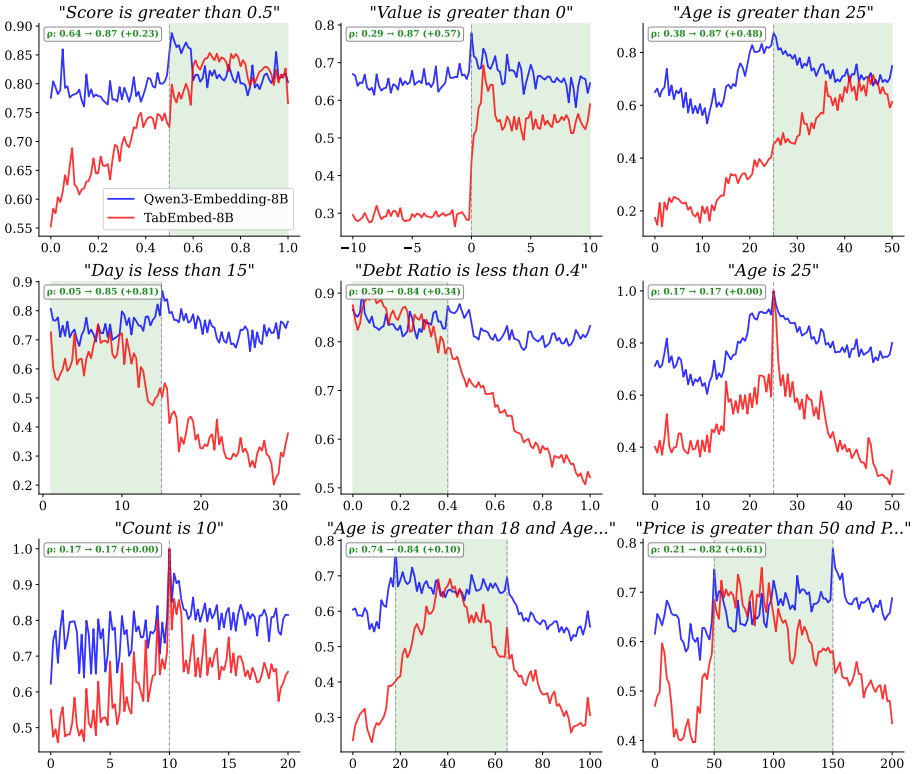

Figure 10: Similarity curves for 9 representative numerical reasoning tasks. **X-axis**: candidate value; **Y-axis**: cosine similarity with the query. **Green shaded regions** indicate valid ranges where conditions are satisfied. **Blue lines**: baseline Qwen3-Embedding-8B; **Red lines**: TabEmbed-8B. Spearman correlation (ρ) improvements are annotated in each subplot.

Audit Level	Matches
Benchmark sets with fuzzy schema match	36 / 311 (11.6%)
Benchmark sets with strict schema match	19 / 311 (6.1%)
Training tables colliding with a schema	33 / 1,750 (1.89%)
Row-level near-duplicates (MinHash)	**0 / 1,750 (0.00%)**
Verified-clean benchmark datasets	275 / 311

Table 7: Train-test overlap audit between the 1,750 training tables and the 311 TabBench classification datasets. Schema-level matches are deliberately conservative; the row-level test finds no near-duplicate record.

Model	Full (311)	Clean (275)
Qwen3-Embedding-0.6B	44.92	43.85
TabEmbed-0.6B	65.27	63.63
Margin	**+20.35**	**+19.78**

Table 8: Overall score on the full benchmark versus the verified-clean subset, which excludes every dataset carrying a schema flag. The margin over the backbone is preserved.

umn counts to be equal. Since schema agreement does not imply content overlap, we further apply a row-level test: for each colliding pair we canonicalize every row into a sorted `column=value` string with numeric values rounded to two decimals, shingle it into character 8-grams, and estimate Jaccard similarity using 128-permutation MinHash, sampling at most 2,000 rows per table and flagging any pair above 0.8.

Table 7 summarizes the outcome. Manual inspection confirms that the schema-level hits are conservative false positives driven by uninformative headers. The `banknote` and `blood-transfusion` datasets expose anonymized column names (`V1-V4`, `Class`) that are subsumed by any training table containing a `V1-V10` range; several finance datasets match on the generic OHLCV schema (`Date`, `Open`, `High`, `Low`, `Close`, `Volume`) shared by essentially every price series on the web; and `adult` matches on column names while sharing no rows. Consistently with this reading, the row-level test returns no near-duplicate pair at all: the training corpus and the benchmark share column vocabulary, not records.

As an independent check, we re-evaluate on the 275 datasets and 25,726 queries that carry no schema flag whatsoever. Table 8 shows that the

margin over the backbone is essentially unchanged (19.78 versus 20.35 points on the full benchmark), confirming that the reported advantage does not originate from train-test overlap.

D Inference Efficiency Analysis

To assess the practical viability of TabEmbed for real-world deployment, particularly in resource-constrained environments, we conduct a comprehensive analysis of the trade-off between model performance and inference latency. Figure 9 illustrates the relationship between the aggregate performance on TabBench (Y-axis) and the computational cost (X-axis) across different model scales.

We measured inference latency using a standardized benchmarking protocol on a single PPU-810E accelerator. To simulate realistic input distributions comparable to those found in TabBench, we constructed a synthetic dataset comprising serialized tabular rows with lengths varying uniformly between 50 and 200 words. All models were evaluated under identical conditions: a batch size of 64 and a maximum sequence length of 1024 tokens. To ensure statistical stability, we performed a warm-up phase followed by three independent experimental runs, reporting the average latency normalized per 10,000 samples.

The results reveal distinct performance-efficiency clusters corresponding to parameter scales. In the low-latency regime, standard text embedding models such as Jina-Embeddings-v3 and Qwen3-Embedding-0.6B offer high throughput but demonstrate limited capability in capturing tabular semantics, with performance scores hovering around 45%. **TabEmbed-0.6B** significantly disrupts this trend, achieving a performance score of 65.27% while maintaining a highly efficient latency profile ($\approx$ 94 seconds per 10k samples). This indicates that domain-specific contrastive learning can unlock tabular reasoning capabilities in lightweight architectures without incurring additional inference costs.

In the high-capacity regime (4B and 8B parameters), TabEmbed continues to push the performance boundary, reaching up to 71.62% with the 8B variant. However, this performance gain comes with a considerable increase in computational cost, with latency exceeding 1,000 seconds per 10k samples. TabEmbed-4B offers a compelling middle ground, delivering near-peak performance at approximately half the inference cost of the 8B model. The plot also includes a theoretical "Oracle" point, highlighting the gap that remains between current state-of-the-art models and an ideal system with minimal delay and maximum accuracy. This suggests that future research directions should focus on knowledge distillation or quantization techniques to retain the structural reasoning capabilities of TabEmbed-8B within the latency budget of smaller models.

E Numeric Sensitivity Curves

To provide a granular view of the numerical reasoning capabilities discussed in Section 5.2, Figure 10 displays the detailed cosine similarity trajectories for nine representative test cases. For each test case, we define a query q containing a specific numerical constraint (e.g., *"Age is greater than 25"*) and generate a sequence of 101 candidate documents $d(x)$ with values linearly spaced across a relevant range. We then compute the cosine similarity score regarding the logical truth value: ideal embeddings should yield high similarity only when the condition is met (indicated by the green shaded regions).

As illustrated by the blue lines in Figure 10, the baseline Qwen3-Embedding-8B typically exhibits random fluctuations or weak correlations. For instance, in *"Score is greater than 0.5"*, the baseline's similarity scores remain relatively flat or erratic regardless of x, confirming that standard text embeddings treat numbers primarily as independent tokens without inherent ordinal semantics.

In contrast, TabEmbed-8B (red lines) demonstrates distinct, logic-aware behaviors tailored to the specific operator types:

- **Inequalities ($>, <$):** The model approximates a step function with sharp transitions at the decision boundary. For example, in *"Age is greater than 25"*, the similarity rises abruptly as x approaches the threshold and sustains a high plateau within the valid range, whereas for *"Day is less than 15"*, it drops significantly once the threshold is exceeded.

- **Equalities ($=$):** For exact matching tasks like *"Age is 25"* or *"Count is 10"*, TabEmbed produces a sharp peak centered exactly at the target value, mimicking a Dirac delta function to distinguish the target from numerically adjacent distractors.

- **Composite Ranges (Between):** For queries involving logical conjunctions (e.g., *"Age is greater than 18..."*), the model accurately delin-

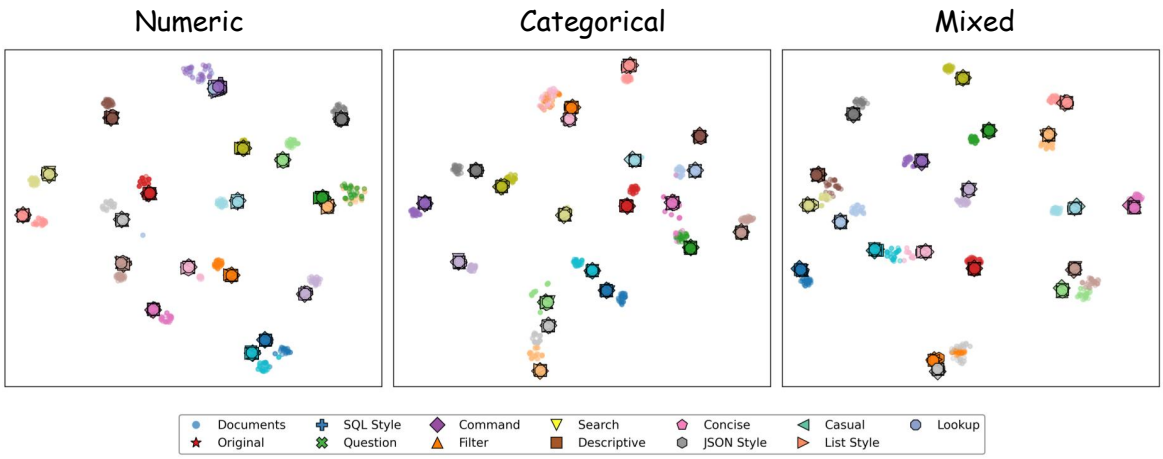

Figure 11: t-SNE visualization of query template robustness across Numeric, Categorical, and Mixed tasks. For each semantic intent, we generate 12 distinct query variations (e.g., SQL-style, JSON-style, Casual) as defined in Table 9. The visualization demonstrates that despite significant syntactic differences, all query variations (represented by distinct markers) cluster tightly around the same relevant documents (colored circles), indicating that TabEmbed learns a syntax-agnostic representation of tabular constraints.

eates the intersection interval, maintaining high similarity only where both conditions hold true.

The substantial improvements in Spearman correlation (ρ) annotated in each subplot (e.g., $0.64 \rightarrow 0.87$) quantitatively verify that TabEmbed has successfully aligned its embedding space with the underlying mathematical logic.

F Robustness to Query Template Variations

A critical requirement for a generalist tabular embedding model is the ability to understand the underlying user intent regardless of the input format. Users may express the same retrieval constraint through diverse modalities, ranging from formal syntaxes (e.g., SQL, JSON) to unstructured natural language (e.g., questions, commands). To evaluate whether TabEmbed has learned a **syntax-agnostic representation** of tabular constraints, we conducted a qualitative visualization experiment. We selected representative samples from the Numeric, Categorical, and Mixed retrieval tasks. For each sample, we generated 12 distinct variations using the templates listed in Table 9, effectively creating a set of semantic equivalence classes where queries differ in surface form but share identical logical constraints.

Figure 11 presents the t-SNE projection of these embeddings. The visualization reveals a striking geometric pattern: for every semantic intent, the diverse query variations (represented by distinct markers such as stars, crosses, and triangles) form tight, cohesive clusters surrounding their corresponding ground-truth documents (colored circles). Notably, this alignment persists across extreme syntactic disparities. For instance, highly structured formats like **JSON Style** (T9: "age": 25...) and **SQL Style** (T2: SELECT * FROM...) are mapped to the immediate vicinity of unstructured natural language queries like **Casual** (T10) and **Question** (T3). This observation confirms that TabEmbed does not merely rely on keyword matching or rigid template overfitting. Instead, it has successfully learned to extract the **invariant logical semantics** (e.g., numerical magnitude and equality constraints) from the input, projecting semantically equivalent queries to the same point on the manifold regardless of their linguistic style. This capability ensures that TabEmbed can generalize to diverse real-world search scenarios where user querying habits may vary significantly.

Quantitative Generalization to Unseen Styles The visualization above is qualitative. To quantify style robustness, we re-render all 30,000 test queries in each of the twelve templates of Table 9 and re-run retrieval with the document corpus and the ground-truth relevance sets held fixed, so that only the surface form of the query changes. Training uses the T1 phrasing exclusively; templates T2-T12 are therefore never observed during training.

As reported in Table 10, TabEmbed degrades by only 2.3 points when moving from the seen template to the eleven unseen ones (65.6 to 63.3), and

ID	Template Name	Example Query
T1	Original	Find records where age is 25 and salary greater than 50000
T2	SQL Style	SELECT * FROM table WHERE age = 25 AND salary > 50000
T3	Question	Which records have age equal to 25 and salary greater than 50000?
T4	Command	Get all entries with age of 25 and salary above 50000
T5	Filter	Filter: age==25, salary>50000
T6	Search	Search for records: age:25 salary:>50000
T7	Descriptive	I need data where the age is 25 and the salary is more than 50000
T8	Concise	age == 25 | salary > 50000
T9	JSON Style	{"age": 25, "salary": {"$gt": 50000}}
T10	Casual	Show me rows that have age as 25 and salary over 50000
T11	List Style	Conditions: 1. age equals 25; 2. salary > 50000
T12	Lookup	Look up records matching: age=25, salary>50000

Table 9: Query template variations used to evaluate semantic robustness of embedding models. All templates express the same underlying constraints but differ in linguistic style and format.

Query Template	Qwen3-0.6B	TabEmbed-0.6B	Δ
T1 Original (seen)	30.6	65.6	+35.0
T2 SQL Style	37.5	66.9	+29.4
T3 Question	32.2	66.5	+34.3
T4 Command	25.7	60.0	+34.3
T5 Filter	22.0	63.0	+41.0
T6 Search	23.1	59.1	+36.0
T7 Descriptive	35.3	67.3	+32.0
T8 Concise	28.6	61.7	+33.1
T9 JSON Style	22.0	60.8	+38.8
T10 Casual	29.3	64.9	+35.6
T11 List Style	30.0	63.5	+33.5
T12 Lookup	26.7	62.9	+36.2
Unseen average	**28.4**	**63.3**	**+34.9**

Table 10: Retrieval performance (nDCG@10) under query template shift. Only T1 is observed during training; T2-T12 are unseen phrasings of the same 30,000 queries evaluated against an unchanged corpus.

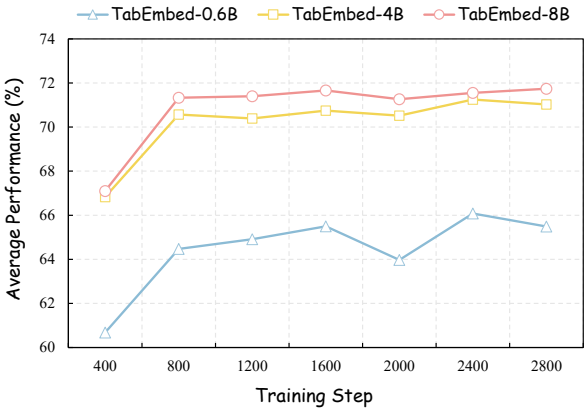

Figure 12: The impact of training steps on the average performance (the macro-average of the four evaluation metrics) across TabBench. We track the evaluation metrics of TabEmbed at different parameter scales (0.6B, 4B, and 8B) from 400 to 2800 training steps.

the backbone degrades in parallel by 2.2 points (30.6 to 28.4). The margin exceeds 34 points on average and remains above 29 points on every individual template, with the weakest absolute case (T6 Search, 59.1) still 36.0 points above the backbone. A gap produced by familiarity with the training phrasing would be expected to collapse under this shift; it does not, indicating that the improvement reflects tabular understanding rather than memorization of surface form.

G Training Convergence Analysis

To determine the optimal training duration and investigate the convergence behavior of our unified contrastive learning paradigm, we monitor the average performance of TabEmbed across three parameter scales (0.6B, 4B, and 8B) at regular intervals, ranging from 400 to 2800 training steps.

As illustrated in Figure 12, we observe several key findings regarding training efficiency and model scaling. First, Rapid Convergence. All models exhibit a steep performance trajectory in the initial phase. Notably, a significant portion of the performance gain is realized within the first 800 steps. For instance, the 4B model improves from roughly 67% to over 70% in this short period. The performance curves generally plateau after approximately 1600 steps, suggesting that our framework is highly data-efficient and does not require excessively prolonged training to learn robust tabular

Algorithm 2 Dynamic Target Identification and Discretization

Input: A tabular dataset $\mathbf{T}$ with columns $\mathcal{C} = \{c_1, c_2, \ldots, c_M\}$, categorical sampling probability p (e.g., $p = 0.5$).
Output: Selected target column y and its processed discrete labels.

Initialize valid candidate set $\mathcal{V} \leftarrow \emptyset$.
for each column $c \in \mathcal{C}$ **do**
  $U_c \leftarrow$ UniqueValues(c)
  % Check against all rejection criteria
  if Name(c) contains "Unnamed:" **or** Type(c) is Date/Time **then**
    continue
  else if $|U_c| < 2$ **or** $|U_c| > 50$ **or** $\max_{v \in U_c}$ Length$(v) > 256$ **then**
    continue
  else if $|U_c| ==$ Rows$(\mathbf{T})$ **and** c is not strictly numeric **then**
    continue
  else
    $\mathcal{V} \leftarrow \mathcal{V} \cup \{c\}$
  end if
end for
Partition $\mathcal{V}$ into numerical candidates $\mathcal{V}_{\text{num}}$ and categorical candidates $\mathcal{V}_{\text{cat}}$.
if $\mathcal{V}_{\text{num}} \neq \emptyset$ **and** $\mathcal{V}_{\text{cat}} \neq \emptyset$ **then**
  Sample y uniformly from $\mathcal{V}_{\text{cat}}$ with probability p, otherwise from $\mathcal{V}_{\text{num}}$.
else
  Sample y uniformly from $\mathcal{V}$.
end if
if $y \in \mathcal{V}_{\text{num}}$ **then**
  $\mathcal{Q} \leftarrow$ Calculate 4-quantiles for y
  Discretize continuous values in y into 4 buckets based on $\mathcal{Q}$.
  Convert each bucket into natural language descriptors (e.g., "less than q_1").
end if
return y

representations. Second, Impact of Model Scale. Consistent with neural scaling laws, larger models consistently achieve higher performance ceilings. TabEmbed-8B (pink line) maintains a clear superiority over the 4B and 0.6B variants throughout the training process. Furthermore, model scale correlates positively with training stability. The 4B and 8B models display smooth and monotonic improvements, whereas the 0.6B model (blue line) exhibits noticeable volatility, particularly around step 2000. This suggests that larger foundation models possess a more robust latent space, making them less susceptible to batch noise during contrastive optimization.

H Target Column Selection Criteria

To transform the unannotated tables from the T4 corpus into high-quality supervised classification tasks, we employ a dynamic target identification pipeline. The complete procedure is summarized in Algorithm 2.

For a given table $\mathbf{T}$, we first consider all columns as potential target candidates and then apply a rigorous filtering protocol to exclude non-informative or trivial prediction targets. Specifically, a column c is excluded from the candidate pool if it satisfies any of the following rejection criteria:

- **Low Informativeness:** The column contains only a single unique value (constant columns), providing no discriminative signal.
- **High Cardinality / Identifiers:** The column possesses a unique value for every row (e.g., UIDs, Row IDs), or the number of unique classes exceeds 50. Such columns typically lead to trivial memorization rather than semantic generalization.
- **Data Type Constraints:** The column is identified as a date/timestamp, or contains textual values exceeding 256 characters, which are unsuitable for standard classification objectives.
- **Column Name Constraint:** The column name contains "Unnamed:" (pandas' default marker for unnamed columns).

It is important to note that columns failing these criteria are only excluded from being selected as the *prediction target* y. They remain part of the input feature set $\mathbf{x}_{-y}$ to provide context, unless they are removed by standard feature selection processes.

Once the set of valid candidate columns is established, we select a single target y for each training instance. To balance the distribution of task types, we employ a weighted sampling strategy. Based on

our qualitative observation that categorical columns often yield higher-quality decision boundaries than arbitrary numerical regression targets, we prioritize classification tasks. Specifically, if both continuous and categorical candidates are present, we sample a categorical target with probability $p = 0.5$ and a continuous target with probability $1 - p = 0.5$.

In cases where a continuous column is selected as the target, we transform the regression problem into a classification problem via dynamic discretization. We divide the continuous range into quantile-based bins (defaulting to 4 buckets) to ensure class balance. The resulting targets are serialized into natural language class descriptors, such as *"less than 15.5"*, *"between 15.5 and 40.2"*, or *"greater than 40.2"*. This unified serialization allows TabEmbed to handle both original categorical labels and discretized numerical bins within the same semantic embedding space.

We compare TabEmbed against a diverse set of state-of-the-art generalist text embedding models, ranging from lightweight encoders to large-scale LLM-based embeddings.

0.6B Parameter Scale

- **Jina-Embeddings-v3** (Sturua et al., 2024): A multilingual, multi-task embedding model based on the Jina-XLM-RoBERTa architecture. It incorporates Rotary Position Embeddings (RoPE) to support extended context windows up to 8192 tokens. A key feature of this model is the integration of five task-specific LoRA adapters, allowing for efficient generation of embeddings tailored to specific downstream applications.

- **Jasper-Token-Compression** (Zhang et al., 2025a): A 600M parameter model from the Jasper series that introduces dynamic text token compression, inspired by DeepSeek-OCR strategies. By combining vector distillation with contrastive learning, it achieves high performance while compressing textual information by approximately 10x, offering a unique approach to efficient representation.

- **Qwen3-Embedding-0.6B** (Zhang et al., 2025b): The lightweight variant of the latest proprietary embedding series from the Qwen family. Building upon the dense Qwen3 foundational architecture, it inherits strong multilingual capabilities and reasoning skills, optimized specifically for retrieval and ranking tasks.

4B Parameter Scale

- **F2LLM-4B** (Zhang et al., 2025c): Standing for "Foundation to Feature Large Language Models," F2LLM is fine-tuned on a curated corpus of 6 million high-quality query-document pairs sourced exclusively from open-source datasets. It employs a single-stage training process with homogeneous macro batches, eschewing complex multi-stage pipelines while covering diverse retrieval and clustering tasks.

- **Octen-Embedding-4B** (Team, 2025): Built upon the Qwen3 foundation, this model is specifically optimized for complex, real-world industry retrieval scenarios. Its training pipeline leverages large-scale domain-specific synthetic data across legal, finance, healthcare, and code domains. By employing parameter-efficient LoRA fine-tuning, cross-device negative sharing, and multi-domain model fusion, it achieves a strong balance between retrieval performance and computational efficiency while supporting ultra-long contexts of up to 32,768 tokens.

- **Qwen3-Embedding-4B** (Zhang et al., 2025b): A mid-sized model in the Qwen3 embedding series. It balances computational efficiency with the advanced long-text understanding capabilities of the Qwen3 foundation, serving as a strong baseline for mid-scale generalist text embeddings.

I Baselines

7B-8B Parameter Scale

- **SFR-Embedding-Mistral** (Meng et al., 2024): Developed by Salesforce Research, this model is initialized from E5-Mistral-7b-instruct and Mistral-7B-v0.1. It represents a robust baseline for instruction-tuned embeddings derived from decoder-only architectures.

- **Linq-Embed-Mistral** (Kim et al., 2024): Also built upon the E5-Mistral and Mistral-7B foundations, Linq-Embed-Mistral focuses on enhancing retrieval performance through advanced data refinement. Its training pipeline emphasizes sophisticated data crafting, rigorous filtering, and hard-negative mining guided by teacher models to improve the quality of synthetic training triplets.

Model	Overall	Accuracy	F1	MRR@10	nDCG@10
Qwen3-Embedding-0.6B	44.92	62.81	50.32	36.00	30.56
SCARF-style	45.07	67.62	58.34	28.92	25.38
TabEmbed-0.6B (Ours)	65.27	67.16	56.56	71.72	65.64

Table 11: Row-to-row contrastive pre-training versus our language-to-row formulation, trained from an identical backbone under a matched budget. Row-to-row training preserves classification quality but collapses retrieval.

- **GTE-Qwen2-7B-Instruct** (Li et al., 2023): The latest addition to the General Text Embedding (GTE) family, built on the Qwen2-7B LLM. It leverages the same training data and strategies as its predecessor (GTE-Qwen1.5) but benefits from the architectural upgrades of the Qwen2 base model. It is a leading performer on the MTEB benchmark, particularly in multilingual evaluation scenarios.

- **Qwen3-Embedding-8B** (Zhang et al., 2025b): The largest model in our comparison suite and the direct backbone for TabEmbed-8B. It represents the state-of-the-art in the Qwen family for dense retrieval, bitext mining, and classification, providing a rigorous baseline to measure the impact of our domain-specific contrastive learning paradigm.

I.1 Row-to-Row Contrastive Baseline

The baselines above are all generalist text embedding models. To isolate the contribution of the language-to-row formulation itself, we additionally train a row-to-row contrastive baseline in the style of SCARF (Bahri et al., 2021) under a matched protocol: the same Qwen3-Embedding-0.6B backbone, the same source tables, the same retrieval-side budget of 600K triplets, and identical optimization hyperparameters. The only difference lies in triplet construction. The anchor is a serialized row, the positive is a re-serialization of the same row under feature dropout ($p = 0.3$), and negatives are obtained with the same mining pipeline and symbolic filters used for TabEmbed. This variant therefore differs from TabEmbed exclusively in what the embedding is asked to align: another view of the same row, rather than a natural language constraint.

Table 11 shows that row-to-row training is genuinely effective for classification: it lifts accuracy from 62.81 to 67.62 and Macro-F1 from 50.32 to 58.34, matching or exceeding TabEmbed on both. Retrieval, however, collapses. nDCG@10 falls to 25.38 against TabEmbed's 65.64, and MRR@10 (28.92) drops below that of the backbone itself (36.00). Aligning augmented views of the same row teaches the model which rows resemble one another, but supplies no signal for aligning a natural language constraint with the rows that satisfy it; the resulting geometry actively degrades the query-to-row alignment already present in the backbone. This substantiates the argument of Section 3.2: an embedding space that serves retrieval as well as classification must be anchored in language rather than in row-level augmentation.

J Applications

The unified representation capability of TabEmbed and the comprehensive evaluation framework of TabBench open up several promising avenues for real-world applications. In particular, it complements existing tabular reasoning systems by providing a foundational, schema-agnostic semantic layer.

J.1 Foundational Retrieval Layer for Agentic RAG Systems

Recent advancements in agentic RAG and Text-to-SQL pipelines have demonstrated strong capabilities in performing complex tabular reasoning, such as aggregations and table joins, when the database schema is well-defined. However, Text-to-SQL approaches face significant challenges in scenarios involving unstructured queries, fuzzy matching, or instances where the user is unaware of the underlying schema. Instead of replacing these reasoning agents, TabEmbed serves as a crucial complementary **Foundational Retrieval Layer**. By mapping natural language constraints (e.g., *"High-value users from the tech sector"*) and structured rows into a shared vector space, it enables databases to perform millisecond-level row retrieval via vector similarity search. This effectively acts as a high-efficiency, cost-effective filter that retrieves relevant context before passing it to downstream LLM agents for heavier logical reasoning.

J.2 Enterprise Data Discovery and Data Lakes

Large enterprises often maintain massive data lakes containing thousands of unorganized spreadsheets and CSV files (the "dark data" problem). In such massive-scale scenarios, running an LLM agent to analyze schemas and generate SQL for every query is computationally prohibitive and incurs unacceptable latency. TabEmbed overcomes this bottleneck by enabling **Offline Indexing and Semantic Data Discovery**. By embedding sample rows or summarized schemas from thousands of tables into a unified vector index, users can search for datasets using vague intent queries (e.g., *"I need sales data regarding Q3 revenue in Southeast Asia"*). Unlike strict keyword-based search or exact SQL matching, TabEmbed understands the numerical and categorical semantics within the table content, efficiently locating relevant tables even if the column headers do not explicitly match the query keywords.

J.3 Zero-Shot and Cold-Start Tabular Prediction

In many dynamic industrial applications (e.g., fraud detection in new markets, user churn prediction for new products), historical training data is scarce or unavailable. Traditional supervised models (like XGBoost) are rendered ineffective in these **Cold-Start Scenarios** because they require task-specific retraining on fixed schemas. As demonstrated by our classification experiments, TabEmbed possesses strong zero-shot transfer capabilities. It can function as a generic feature extractor or a nearest-neighbor classifier right out of the box. Practitioners can simply convert a handful of labeled examples (a support set) into vectors and classify new incoming rows based on embedding similarity, enabling immediate predictive capabilities without the need for time-consuming feature engineering or model training.

J.4 Cross-Schema Entity Resolution and Data Integration

A pervasive challenge in database management is **Entity Resolution** (or Record Linkage), identifying rows across different databases that refer to the same real-world entity despite having different schemas, missing values, or inconsistent naming conventions (e.g., merging a "Client" table with a "Customer" table after a corporate acquisition). Traditional methods rely heavily on manual schema matching and hand-crafted string similarity rules. Because TabEmbed serializes diverse tabular fields into a unified natural language context, it inherently learns a schema-agnostic representation. Two rows describing the same entity with different column names or formatting will be projected into close proximity within the embedding space. Consequently, enterprise data integration can be elegantly reformulated as a cross-database vector similarity search, bypassing the arduous process of manual schema alignment.

J.5 Semantic Anomaly Detection and Data Cleaning

Real-world tabular data is notoriously noisy, often containing logical contradictions (e.g., a "Status: Active" subscription with a "Termination Date" in the past) or numerical errors. Rule-based data cleaning requires domain experts to anticipate and hardcode every possible error type. TabEmbed offers a robust alternative for out-of-the-box **Semantic Anomaly Detection**. By projecting all rows of a table into the learned embedding space, standard density-based anomaly detection algorithms (such as Isolation Forest or Local Outlier Factor) can be directly applied to the dense vectors. Since TabEmbed is pre-trained to understand structural logic and numerical magnitude, rows containing semantic contradictions or extreme outliers will naturally isolate themselves in the manifold. This provides a zero-shot, automated data cleaning mechanism that does not rely on predefined schemas or rules.