TabEmb: Joint Semantic-Structure Embedding for Table Annotation

Ehsan Hoseinzade
School of Computing Science
Simon Fraser University
Burnaby, Canada
ehoseinz@sfu.ca

Ke Wang
School of Computing Science
Simon Fraser University
Burnaby, Canada
wangk@cs.sfu.ca

Anandharaju Durai Raju
School of Computing Science
Simon Fraser University
Burnaby, Canada
aduraira@sfu.ca

Abstract

Table annotation is crucial for making web and enterprise tables usable in downstream NLP applications. Unlike textual data where learning semantically rich token or sentence embeddings often suffice, tables are structured combinations of columns wherein useful representations must jointly capture column's semantics *and* the inter-column relationships. Existing models learn by linearizing the 2D table into a 1D token sequence and encoding it with pretrained language models (PLMs) such as BERT. However, this leads to limited semantic quality and weaker generalization to unseen or rare values compared to modern LLMs, and degraded structural modeling due to 2D-to-1D flattening and context-length constraints. We propose **TabEmb**, which directly targets these limitations by decoupling semantic encoding from structural modeling. An LLM first produces semantically rich embeddings for each column, and a graph-based module over columns then injects relationships into the embeddings, yielding joint semantic–structural representations for table annotation. Experiments show that TabEmb consistently outperforms strong baselines on different table annotation tasks. Source code and datasets are available at https://github.com/hoseinzadeehsan/TabEmb

1 Introduction

Tabular data is widespread and rapidly proliferating structured information on the web and in enterprise data lakes. Robust and scalable understanding of such data automatically is critical for downstream applications including data integration (Hai et al., 2023), data cleaning (Limaye et al., 2010; Kandel et al., 2011), schema matching (Rahm and Bernstein, 2001), data discovery (Fernandez et al., 2018b,a), and knowledge graph construction (Zhong et al., 2023; Steenwinckel et al., 2019). A foundational step in this process is *table annotation*, which assigns semantic labels to columns through Column Type Annotation (CTA) (Zhang et al., 2019; Hulsebos et al., 2019; Suhara et al., 2022; Deng et al., 2020; Hoseinzade and Wang, 2024, 2025, 2026; Zhang et al., 2024b; Korini and Bizer, 2023), to relations between columns through Column Property Annotation (CPA) (Deng et al., 2020; Korini and Bizer, 2024; Suhara et al., 2022; Zhang et al., 2024b), and to entire tables through Table Type Annotation (TTA) (Kayali et al., 2024; Korini and Bizer, 2023, 2024).

While recent works explored LLM-based zero- and few-shot approaches for some of these tasks without task-specific training for minimal reliance on labeled data (Zhang et al., 2024b; Korini and Bizer, 2023, 2024; Kayali et al., 2024; Zhang et al., 2024a), they under-perform compared to supervised approaches on complex CTA/CPA/TTA datasets with large label spaces (Zhang et al., 2024b; Feuer et al., 2024). Within supervised table annotation, recent progress has been driven by improved *column embedding* representations. Effective representations must satisfy two requirements: (i) *high semantic fidelity*, capturing meaning of a column's content, and (ii) *explicit structural modeling*, capturing relationships among table's columns. From this perspective, supervised methods are broadly categorized into three regimes.

Single-column approaches (Chen et al., 2019a; Hulsebos et al., 2019; Feuer et al., 2024; Sun et al., 2023) aim at learning semantics of *individual* columns relying on their statistical features or via textual encoders, however, they ignore intra-table dependencies that are often essential for disambiguation.

Structure-augmented approaches (Zhang et al., 2019; Hoseinzade and Wang, 2024) address dependencies between columns, but typically as a post-hoc refinement over prediction outputs (logits) and serve only a "structure-blind prediction" rather than learning "structure-aware" embeddings.

Joint modeling approaches (Yin et al., 2020;

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 16620–16631

Deng et al., 2020; Iida et al., 2021; Wang et al., 2021; Suhara et al., 2022; Ding et al., 2025) attempt to jointly capture semantics and structure using pretrained language models (PLMs) like BERT by encoding entire tables. However, three limitations remain. First, they have **limited semantic coverage and world knowledge** compared to modern LLMs, leading to weaker generalization to unseen column values at test time and less well-separated clustering in the column embedding space (Fig. 1a). Second, linearizing tables from 2D to 1D sequences **destroys the native relational structure** and imposes strict input-length limits (e.g., 512 tokens for BERT), degrading performance on wide tables. Third, adapting PLMs to tabular structure typically requires **expensive PLM fine-tuning**.

To address these challenges, we introduce **TabEmb** (Fig 2), a modular framework for learning *joint semantic-structural embeddings (structure-aware embeddings)* for supervised table annotation. TabEmb encodes each column independently using a frozen LLM, yielding semantically rich and well-separable embeddings (Fig. 1b) that generalize better to unseen values and rare semantic types. It then uses a trainable graph neural network (GNN) to perform message passing over a fully-connected graph of columns, injecting intra-table structure directly at the *embedding level*. Finally, the task-specific classification heads operate on these embeddings to support CTA/CPA/TTA. By sharing the frozen LLM backbone across tasks, and training the GNNs independently for the demands of CTA/CPA/TTA tasks, TabEmb combines LLM-level semantic quality with explicit structural modeling producing more discriminative embedding spaces (Fig. 1c) while remaining computationally practical by avoiding expensive full-model fine-tuning.

Our core contributions are as follows:

- We introduce **TabEmb**, a modular framework that combines frozen LLM-based semantic encoding with trainable GNN-based structural modeling to learn high-quality semantic-structural column embeddings (Figure 1).
- We benchmark TabEmb across the CTA, CPA and TTA tasks, six datasets, and 8 baselines. TabEmb outperforms all prior methods in micro-F1 score, achieving average gains of 4.2 on CTA, 4.7 on CPA, and 5.6 on TTA over the strongest baseline for each task (Table 2).
- TabEmb trains significantly faster than fine-tuned LLM baselines by keeping the LLM frozen and training only lightweight GNN and classification heads (Table 3).

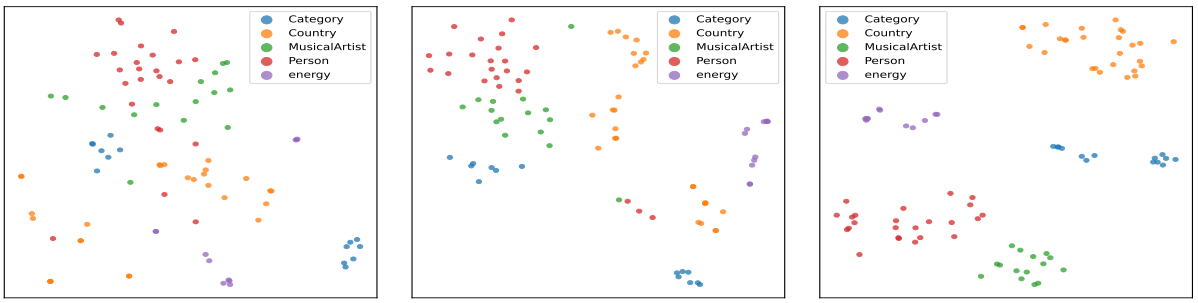
2 Related Work

Recent methods for table annotation can be grouped into four categories by *what* they model:

Single-column models. These early approaches focus on modeling each column independently. ColNet (Chen et al., 2019a) uses DBpedia lookups to create training examples and learns a CNN over column values. Sherlock (Hulsebos et al., 2019) combines hand-crafted statistical and textual features with a feed-forward network. RECA (Sun et al., 2023) leverages BERT with retrieved inter-table context to strengthen per-column representations. ArcheType (Feuer et al., 2024) fine-tunes an open-source LLM on column-type labels. These methods emphasize semantic quality of columns' representation but ignore intra-table dependencies.

Structure-augmented models. These approaches exploit relationships between columns given initial predictions. SATO (Zhang et al., 2019) extends Sherlock with a CRF over adjacent columns to enforce local dependencies. GAIT (Hoseinzade and Wang, 2024) applies a GNN over RECA's column-level predictions to capture intra-table dependencies. In both cases, structure operates as a refinement layer *on top of* prediction of another model, rather than being integrated into the embeddings.

Joint semantic-structural models. Joint models aim to encode both semantics of column content and table structure in a unified representation. HNN (Chen et al., 2019b) models intra-column semantics, enhancing ColNet. With pretrained language models, several methods feed an entire table (or large portions of it) to a PLM like BERT-style encoders and fine-tune the model. TaBERT (Yin et al., 2020) utilizes BERT as a base model to capture the table content features; TURL (Deng et al., 2020) is pre-trained for table understanding and then fine-tuned for column type annotation; TABBIE (Iida et al., 2021) separately processes the rows and columns of tables to give a better understanding of them; TCN (Wang et al., 2021) suggests using both intra-table and inter-table information for column type prediction; Doduo (Suhara et al., 2022) predicts all the columns of a table together by feeding the whole table to BERT; KGLink (Wang et al.,



(a) PLM (BERT) (b) LLM (Mistral) (c) TabEmb (Mistral + Trainable GNN)

Figure 1: **The Impact of joint semantic-structure Modeling.** t-SNE visualization of column embeddings from three paradigms: (Left) **PLM (BERT)**, (Middle) **LLM (Mistral)**, and (Right) **TabEmb (Mistral + Trainable GNN)** on the $\text{SOTAB}_{\text{dbp}}$ dataset. For readability, we visualize only a subset of classes. While BERT produces weakly separated clusters and the LLM improves semantic coherence, the LLM still exhibits significant overlap between ambiguous types (e.g., *Person* vs. *MusicalArtist*) due to its lack of structural context. By explicitly modeling table structure via a GNN, TabEmb resolves this ambiguity, yielding the most discriminative embedding space.

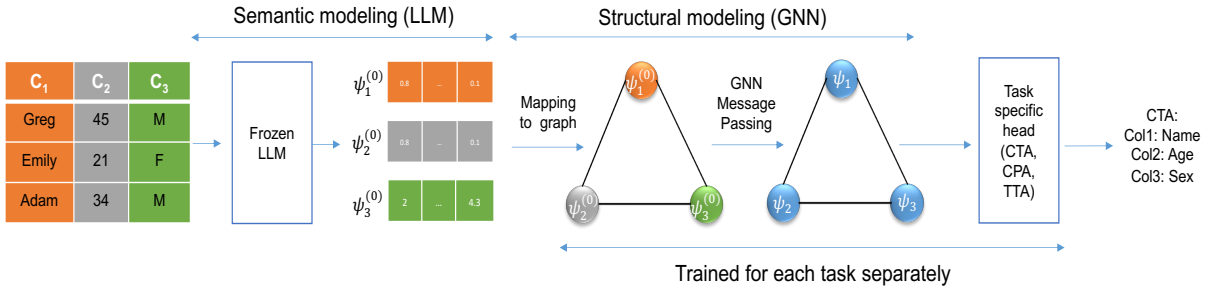


Figure 2: Overview of TabEmb. A frozen LLM encodes each column into a semantic embedding, a GNN refines these via message passing over the column graph, and a lightweight task-specific head predicts labels, with the GNN and head trained separately for each task (CTA/CPA/TTA).

2024) augments BERT with Wikidata to improve type predictions. REVEAL (Ding et al., 2025) selectively chooses the most relevant columns to the target column(s) before feeding them to BERT for prediction. These models rely on expensive fine-tuning of PLMs such as BERT: tables must be linearized into 1D sequences, context length is limited, and their semantic coverage lags behind modern LLMs.

LLM-based prompting approaches. Recent work explores zero- and few-shot table annotation either by prompting general-purpose LLMs (Korini and Bizer, 2023, 2024; Zhang et al., 2024b; Kayali et al., 2024) without supervised training, or by using synthetic supervision generated by LLMs (Hoseinzade and Wang, 2026, 2025), making them complementary to our supervised setting.

3 Problem Definition

Table annotation includes three independent classification tasks: Column Type Annotation (CTA), Column Property Annotation (CPA), and Table Type Annotation (TTA), each involving different prediction targets and using a separate labeled dataset. Importantly, all tasks operate on tables without column headers, only the cell values are provided as input.

Formally, each task is defined over a dataset $D \in \{D_{\text{CTA}}, D_{\text{CPA}}, D_{\text{TTA}}\}$, where both the tables and label sets can differ across tasks. Each table $t = (c_1, c_2, \ldots, c_n)$ consists of n columns and multiple rows of cell values. Each task follows a two-stage learning formulation. First, a feature extraction function ϕ is learned that takes the full table as input and produces a sequence of structure-aware column embeddings:

$$\psi = \phi(t) = \langle \psi_1, \psi_2, \ldots, \psi_n \rangle, \quad \psi_i \in \mathbb{R}^d$$

where ψ_i is the embedding for column c_i. Then, a task-specific classifier f is learned to map the relevant subset of ψ to a label, depending on the task:

- **CTA:** Given a column c_i, predict its semantic type from a predefined, task-specific label set

T_{CTA}:

$$f_{\text{CTA}}(\psi_i) \rightarrow T_{\text{CTA}}$$

- **CPA:** Given a column pair (c_i, c_j), predict their relationship type from a predefined label set T_{CPA}:

$$f_{\text{CPA}}(\psi_i, \psi_j) \rightarrow T_{\text{CPA}}$$

- **TTA:** Given a full table t, predict its category from a predefined label set T_{TTA}:

$$f_{\text{TTA}}(\psi_1, \ldots, \psi_n) \rightarrow T_{\text{TTA}}$$

We treat each task independently, using separate datasets and training separate models. In this supervised setting, for each task we learn both the feature extractor ϕ and the task-specific classifier f_{task} on its own labeled dataset $D^{\text{train}}_{\text{CTA}}$, $D^{\text{train}}_{\text{CPA}}$, and $D^{\text{train}}_{\text{TTA}}$, respectively.

4 TabEmb

TabEmb operates with three components: a frozen large language model $\mathcal{M}_{\text{LLM}}$ that produces semantic embeddings for each column, a graph neural network $\mathcal{M}_{\text{GNN}}$ that refines these embeddings on the column graph into structure-aware column representations, and task-specific heads g_{task} that map the resulting representations to CTA, CPA, or TTA labels.

In the training phase, $\mathcal{M}_{\text{LLM}}$ is kept frozen, and the parameters of $\mathcal{M}_{\text{GNN}}$ and g_{task} are learned end-to-end from labeled tables for each task. In the prediction phase, the trained $\mathcal{M}_{\text{GNN}}$ and g_{task} are used together with $\mathcal{M}_{\text{LLM}}$ to annotate unseen tables. The following subsections describe these training and prediction procedures in detail.

4.1 Training

The training phase of TabEmb, described in Algorithm 1, is run *separately* for each task task $\in \{\text{CTA}, \text{CPA}, \text{TTA}\}$. For each training table $(t, y) \in D^{\text{train}}_{\text{task}}$, the label y is defined according to the task: for CTA, it contains one semantic type per column; for CPA, it specifies a relation type for each column pair; and for TTA, it assigns a single type to the entire table. These labels guide the end to end supervised training of the GNN $\mathcal{M}_{\text{GNN}}$ and task-specific classifier g_{task}, which consists of three phases:

Module initialization. For a given task (CTA, CPA, or TTA), we instantiate the corresponding prediction head g_{task} via **TaskClassifier(task)**, which selects a per-column, per-pair, or pooled-table classifier with the appropriate input–output shape. We also initialize a trainable graph neural network $\mathcal{M}_{\text{GNN}}$ that refines LLM-based embeddings using structural relationships between table columns.

Algorithm 1 Training Phase of TabEmb

Input: Task task $\in \{\text{CTA}, \text{CPA}, \text{TTA}\}$, Training dataset $D^{\text{train}}_{\text{task}}$, frozen LLM $\mathcal{M}_{\text{LLM}}$, trainable GNN $\mathcal{M}_{\text{GNN}}$
Output: Trained GNN $\mathcal{M}_{\text{GNN}}$, classifier g_{task}

{*Module initialization*}
$g_{\text{task}} \leftarrow$ **TaskClassifier**(task)
{*Semantic Embedding and Graph Construction*}
$\mathcal{G} \leftarrow \emptyset$
for each $(t, y) \in D^{\text{train}}_{\text{task}}$ **do**
  $\psi^{(0)} \leftarrow$ **ColumnEmbedding**$(t, \mathcal{M}_{\text{LLM}})$
  $G_t \leftarrow$ **ConstructGraph**$(t, \psi^{(0)})$
  Add (G_t, y) to $\mathcal{G}$
end for
{*Learning structure-aware embeddings and classifier*}
for each training epoch **do**
  for each mini-batch $\{(G_b, y_b)\}_{b=1}^{B} \subset \mathcal{G}$ **do**
    $\mathcal{L} \leftarrow 0$
    for b = 1 to B **do**
      $\psi \leftarrow$ **StructEmbedding**$(G_b, \mathcal{M}_{\text{GNN}})$
      $\hat{y} \leftarrow g_{\text{task}}(\psi)$
      $\mathcal{L} \leftarrow \mathcal{L} + \text{Loss}(\hat{y}, y_b)$
    end for
    Update $\mathcal{M}_{\text{GNN}}, g_{\text{task}}$ using gradient descent on $\mathcal{L}/B$
  end for
end for
return $\mathcal{M}_{\text{GNN}}, g_{\text{task}}$

Semantic embedding and graph construction. For each labeled table $(t, y) \in D^{\text{train}}_{\text{task}}$, we generate semantic embeddings for each column using the frozen LLM $\mathcal{M}_{\text{LLM}}$. This is handled by **ColumnEmbedding**$(t(c_1, ..., c_n), \mathcal{M}_{\text{LLM}})$, which returns initial column embeddings $\psi^{(0)} = (\psi^{(0)}_1, \ldots, \psi^{(0)}_n)$. These embeddings are then used to construct a graph G_t, where each column becomes a node initialized with its embedding, via **ConstructGraph**$(t(c_1, ..., c_n), \psi^{(0)})$. We precompute and store each labeled graph (G_t, y) in a graph pool $\mathcal{G}$; frozen LLM is not used during subsequent gradient-based training.

Learning structure-aware embeddings and classifier. After preprocessing, we train the GNN and classifier end-to-end over the graph pool. We apply **StructEmbedding**$(G_b, \mathcal{M}_{\text{GNN}})$ to each graph G_b in the batch of labeled graphs $\{(G_b, y_b)\}_{b=1}^{B} \subset \mathcal{G}$ to refine the column embeddings via message passing, obtaining structure-aware embeddings ψ. These are passed to the task-specific classifier g_{task} to generate predictions $\hat{y}$. We compute the loss $\mathcal{L} = \text{Loss}(\hat{y}, y_b)$ over the batch of graphs and update the parameters of $\mathcal{M}_{\text{GNN}}$ and g_{task} via gradient descent.

We define the core functions in Algorithm 1:

TaskClassifier(task): Returns the task-specific classifier g_{task}: a per-column head for CTA, a per-pair head for CPA, or a per-table classifier for TTA. It is implemented as a simple linear mapping (single output projection, no hidden layers) from the learned embedding space to the task label space.

ColumnEmbedding(t, $\mathcal{M}_{\text{LLM}}$): For each column c_i in the table $t = (c_1, \ldots, c_n)$, we uniformly sample m non-null cell values, concatenate them into a text sequence, and feed it to the frozen LLM $\mathcal{M}_{\text{LLM}}$. We then mean-pool the output token representations, i.e., take the average of all token embeddings, to obtain an initial column embedding $\psi_i^{(0)} \in \mathbb{R}^d$, which we find more stable than using only the last token embedding. Repeating this for all columns yields the set of initial embeddings $\psi^{(0)} = (\psi_1^{(0)}, \ldots, \psi_n^{(0)})$.

ConstructGraph(t, $\psi^{(0)}$): Builds a fully connected undirected graph $G_t = (V_t, E_t)$ for table t, where each column c_i corresponds to a node $v_i \in V_t$ initialized with its LLM-based embedding $h_i^{(0)} = \psi_i^{(0)}$. The graph includes both inter-column edges and self-loops to support information propagation and residual modeling. We use a fully connected column graph to avoid assuming prior knowledge about which column pairs are related. This allows the GNN to learn which neighboring columns are most informative and to weight their contributions accordingly. For very wide tables, where a fully connected graph may be costly, we can partition columns into coherent blocks (e.g., 10–15 columns via schema blocks) and construct a fully connected graph within each block.

StructEmbedding($G_b, \mathcal{M}_{\text{GNN}}$): Applies a multi-layer GNN $\mathcal{M}_{\text{GNN}}$ (e.g., a Graph Attention Network (Veličković et al., 2017)) to G_b, refining node representations through several rounds of message passing with residual connections. Starting from the initial node embeddings $h_i^{(0)}$, the GNN layers iteratively update all nodes, and after S layers we obtain the final semantic–structure column embeddings $\psi = (\psi_1, \ldots, \psi_n)$, where each $\psi_i \in \mathbb{R}^d$ is the refined embedding of column i in the table.

4.2 Prediction

At prediction time (Algorithm 2), TabEmb follows the same pipeline used during training. Given a new input table $t = (c_1, \ldots, c_n)$, the model first encodes each column independently using the frozen LLM to obtain initial embeddings $\psi^{(0)}$. These embeddings are used to construct a fully connected graph G_t, where each node represents a column and is initialized with its embedding.

The trained GNN $\mathcal{M}_{\text{GNN}}$ is then applied to refine the embeddings through message passing over the graph, resulting in structure-aware representations ψ. Finally, the task-specific classifier g_{task} consumes these refined embeddings to produce the output $\hat{y}$, according to the task.

Algorithm 2 Prediction Phase of TabEmb

Input: Table $t = (c_1, \ldots, c_n)$, trained GNN $\mathcal{M}_{\text{GNN}}$, trained classifier g_{task}, frozen LLM $\mathcal{M}_{\text{LLM}}$
Output: Predicted output $\hat{y}$ for task task $\in$ {CTA, CPA, TTA}

$\psi^{(0)} \leftarrow$ **ColumnEmbedding**$(t, \mathcal{M}_{\text{LLM}})$
$G_t \leftarrow$ **ConstructGraph**$(t, \psi^{(0)})$
$\psi \leftarrow$ **StructEmbedding**$(G_t, \mathcal{M}_{\text{GNN}})$
$\hat{y} \leftarrow g_{\text{task}}(\psi)$
return $\hat{y}$

5 Evaluation

We evaluated TabEmb on the tasks of CTA, CPA, and TTA. All the experiments were run on a NVIDIA RTX 6000 Ada GPU.

5.1 Method

Datasets. We considered six datasets covering diverse domains and ontologies: T2D (Chen et al., 2019b; t2d, 2017; Korini and Bizer, 2024), Wikitable (Deng et al., 2020), Webtables (Suhara et al., 2022; Zhang et al., 2019; Hoseinzade and Wang, 2024), and the three SOTAB-V2 datasets: $\text{SOTAB}_{\text{sch}}$, $\text{SOTAB}_{\text{sch-s}}$, and $\text{SOTAB}_{\text{dbp}}$ (sot, 2023). CTA is evaluated on all six datasets, CPA on all except Webtables, and TTA on the three SOTAB variants and T2D. More details on the choices of datasets are provided in Appendix A. Across all datasets and tasks, we adopt the same train/test/-validation splits or 5-fold cross-validation settings used in prior works.

Metric. Following previous works (Feuer et al., 2024; Hoseinzade and Wang, 2025, 2026), we evaluate the performance using *micro-F1 score* collected on test tables, expressed in percentage.

Default LLM and GNN of TabEmb. We use a frozen Mistral-7B (Jiang et al., 2023) ($\mathcal{M}_{\text{LLM}}$) via HuggingFace to compute column embeddings through sampling m=25 non-null cell values per column. Specifically, we use the HuggingFace *base* model `mistralai/Mistral-7B-v0.1` (not an Instruct variant). A GAT (Veličković et al., 2017)

	Single-column			Structure-augmented		Joint semantic-structure			
	Sherlock	RECA	ArcheType	SATO	GAIT	TURL	Doduo	REVEAL	**TabEmb (Ours)**
CTA	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
CPA	Yes	No	Yes	No	No	Yes	Yes	Yes	Yes
TTA	Yes	No	Yes	No	No	Yes	Yes	No	Yes

Table 1: Different models categorized by how they model semantics and structure: single-column (Sherlock (Hulsebos et al., 2019), RECA (Sun et al., 2023), ArcheType (Feuer et al., 2024)), structure-augmented (SATO (Zhang et al., 2019), GAIT (Hoseinzade and Wang, 2024)), and joint semantic-structure (TURL (Deng et al., 2020), Doduo (Suhara et al., 2022), REVEAL (Ding et al., 2025)), and our TabEmb.

	Single-column			Structure-augmented		Joint semantic-structure			
	Sherlock	RECA	ArcheType	SATO	GAIT	TURL	Doduo	REVEAL	**TabEmb (Ours)**
CTA	77.1	83.4	82.8	77.9	84.7	83.3	85.8	86.5	**90.7**
CPA	50.7	N/A	85.2	N/A	N/A	81.3	84.7	84.4	**89.9**
TTA	47.8	N/A	90.1	N/A	N/A	82.0	86.5	N/A	**95.7**

Table 2: Main results (micro-F1): baselines grouped by how they model semantics and structure.

Model	**AVG. train**	**AVG. test**
Doduo	104.8	0.06
ArcheType	585.9	0.35
TabEmb	**92.1**	0.21

Table 3: Average training time (minutes) and per-table test time (seconds) over all tasks and datasets.

with 2 hidden layers and 4 attention heads ($\mathcal{M}_{\text{GNN}}$) is implemented in DGL (Wang et al., 2019) and trained with Adam (learning rate $1e{-}3$, weight decay $5e{-}4$) for 100 epochs with a batch size of 256. We select the checkpoint with the best validation performance when a validation split is available, and otherwise use the final epoch. Unless otherwise stated, these settings remain consistent for TabEmb across all experiments. Other types of LLMs and GNNs are investigated in Section 5.6.

Baselines. We compare TabEmb against eight *supervised* baselines, grouped by how they model semantics and structure (Table 1). ArcheType is included only in its supervised regime. The zero-/few-shot LLM prompting methods (Zhang et al., 2024b; Korini and Bizer, 2023, 2024; Xiao et al., 2025) are not directly comparable to our supervised, content-only setting.

Single-column. Sherlock, RECA, and ArcheType all support CTA. We adapt Sherlock and ArcheType by concatenating column pairs for CPA or all columns for TTA into a single input; RECA's inter-table, single-column design does not extend beyond CTA.

Structure-augmented. SATO and GAIT are designed for CTA-only structure modeling, and their architectures do not naturally generalize to CPA or TTA.

Joint semantic–structure. TURL, Doduo, and REVEAL all support CTA and CPA; for TTA, we adapt TURL and Doduo via averaged column embeddings, while REVEAL's context-selection per column mechanism does not extend cleanly.

5.2 Main Results

Table 2 reports the average micro-F1 scores for CTA, CPA, and TTA across all applicable datasets. **Bold** indicates the best model in each task. Task-specific results are reported in the following Sections 5.3, 5.4, 5.5.

On CTA, TabEmb reaches 90.7 micro-F1, outperforming the strongest baseline REVEAL's 86.5 micro-F1 by +4.2 points. On CPA, TabEmb attains 89.9 micro-F1, outperforming the strongest baseline ArcheType's 85.2 micro-F1 by +4.7 points. On TTA, TabEmb attains 95.7 micro-F1, outperforming the strongest baseline ArcheType's 90.1 micro-F1 by +5.6 points. Note that this "1-vs-all" comparison compares TabEmb with the best-performing baseline on each task. The "1-vs-1" comparison that compares TabEmb against one baseline at a time on all tasks would yield even larger gains.

Among *single-column* models, ArcheType is the strongest baseline, but it still lags behind TabEmb, indicating that semantics alone are insufficient without explicit structural modeling. GAIT is the best *structure-augmented* model, yet TabEmb performs better because its embedding-level refinement learns richer inter-column relationships than GAIT's prediction-level adjustment, which can only adjust classification scores based on other column's predictions. Doduo and REVEAL are the strongest *joint semantic–structure* baselines and are conceptually closest to TabEmb; however, their BERT-based encoders and tightly coupled

	Single-column			Structure-augmented		Joint semantic-structure			
Dataset	Sherlock	RECA	ArcheType	SATO	GAIT	TURL	Doduo	REVEAL	**TabEmb (Ours)**
$\text{SOTAB}_{\text{sch}}$	80.9	84.9	85.1	82.1	85.8	81.7	86.3	86.6	**90.4**
$\text{SOTAB}_{\text{sch-s}}$	72.6	81.9	83.0	74.2	83.1	75.4	81.1	83.1	**87.5**
$\text{SOTAB}_{\text{dbp}}$	77.5	83.1	83.6	79.0	84.7	76.7	85.2	86.1	**91.0**
T2D	67.7	86.5	88.0	70.7	85.1	94.0	91.1	91.7	**95.5**
Wikitable	75.8	74.2	76.7	76.0	75.5	78.9	75.2	75.8	**82.9**
Webtables	87.9	93.3	80.3	92.5	94.1	93.1	96.4	95.9	**96.8**
Average	77.1	83.4	82.8	79.1	84.7	83.3	85.8	86.5	**90.7**

Table 4: CTA results: micro-F1 score comparison of TabEmb and baseline models.

	Single-column		Joint semantic-structure			
Dataset	Sherlock	ArcheType	TURL	Doduo	REVEAL	**TabEmb (Ours)**
$\text{SOTAB}_{\text{sch}}$	67.6	88.6	83.3	90.2	90.5	**91.7**
$\text{SOTAB}_{\text{sch-s}}$	58.1	87.1	75.2	84.1	85.3	**89.5**
$\text{SOTAB}_{\text{dbp}}$	67.5	89.1	85.7	92.6	91.9	**93.3**
T2D	37.1	70.1	71.8	66.1	63.6	**83.5**
Wikitable	73.6	91.3	90.5	91.3	90.8	**91.5**
Average	50.7	85.2	81.3	84.7	84.4	**89.9**

Table 5: CPA results: micro-F1 score comparison of TabEmb and baseline models.

	Single-column		Joint semantic-structure		
Dataset	Sherlock	ArcheType	TURL	Doduo	**TabEmb (Ours)**
$\text{SOTAB}_{\text{sch}}$	62.0	95.0	85.1	90.5	**97.5**
$\text{SOTAB}_{\text{sch-s}}$	57.6	91.7	81.7	86.4	**96.5**
$\text{SOTAB}_{\text{dbp}}$	49.2	94.2	82.5	86.1	**97.4**
T2D	23.1	79.4	78.4	83.0	**91.5**
Average	47.8	90.1	82.0	86.5	**95.7**

Table 6: TTA results: micro-F1 score comparison of TabEmb and baseline models.

text–structure modeling yield weaker semantic representations and less flexible structure modeling than TabEmb's frozen LLM plus GNN design.

Table 3 reports average training time (in minutes) and average test-time latency per table (in seconds) over all tasks and datasets for models that support all three tasks (excluding TURL and Sherlock due to low performance): Doduo, ArcheType, and TabEmb. TabEmb, despite using a larger LLM backbone, has training time comparable to Doduo because it encodes columns once and then trains only a small GNN and task-specific heads. At test time, latency is dominated by the backbone LLM rather than the GNN or heads, so TabEmb is slower than Doduo but this cost is offset by its higher micro-F1 (+4.9, +5.2, +9.2 on CTA, CPA, and TTA). ArcheType is by far the slowest due to full-model LLM fine-tuning.

Next, we discuss task-specific results.

5.3 CTA

Table 4 presents CTA results across six datasets. TabEmb consistently outperforms all baselines on each dataset. In particular, following the top-down order of the table, TabEmb surpasses the strongest baseline of each dataset by +3.8%, +4.4%, +4.9%, +1.5%, +4.0%, and +0.4%, respectively.

Doduo and REVEAL are usually the strongest baselines, but on very small datasets like T2D (160 training tables) fine-tuning large models with many parameters (ArcheType, Doduo, REVEAL) on tabular inputs that differ from their pre-training objective is difficult and often unstable, so smaller models such as TURL perform relatively better. In contrast, TabEmb keeps the LLM frozen and trains only a GNN and a lightweight classification head, requiring far fewer parameters and achieving the best performance in this low-resource regime.

On the other hand, when there is enough labeled data to adapt models to tabular structure, LLM-based embeddings yield better generalization to unseen values than PLM-based embeddings, reducing reliance on large training sets. This is evident in the comparison between $\text{SOTAB}_{\text{sch}}$ and $\text{SOTAB}_{\text{sch-s}}$, which share the same test and validation splits but differ in training size (44,769 vs. 10,631 tables, respectively). For instance, the BERT-based Doduo and REVEAL drop from 86.3 to 81.1 (-5.2) and from 86.6 to 83.1 (-3.5), while the LLM-based ArcheType and TabEmb go from 85.1 to 83.0 (-2.1) and from 90.4 to 87.5 (-2.9),

respectively. Notably, TabEmb trained on the smaller $\text{SOTAB}_{\text{sch-s}}$ still outperforms the best baseline trained on the larger $\text{SOTAB}_{\text{sch}}$ (87.5 vs. 86.6), as its LLM-based column embeddings generalize better to unseen values.

5.4 CPA

Table 5 presents the CPA results across five datasets. TabEmb consistently outperforms all baselines on each dataset. Again, following the top-down order of the table, TabEmb surpasses the strongest baseline of each dataset by +1.2%, +2.4%, +0.7%, +11.7%, and +0.2%, respectively. Similar to CTA, all models perform worse on the low-resource T2D.

Excluding T2D, the performance gap between TabEmb and strongest baselines (Doduo, ArcheType) is smaller on CPA than on CTA. CPA relies more on the row-wise value alignment between column pairs (e.g., subject–object patterns), which Doduo and ArcheType capture via signals from their row-level table access. TabEmb instead encodes whole column semantics with LLM embeddings and models inter-column dependencies via a GNN, prioritizing global semantic structure, which narrows the relative improvement on CPA.

5.5 TTA

Table 6 reports TTA results on four datasets. TabEmb consistently outperforms all baselines, surpassing the strongest model on each dataset by +2.5%, +4.8%, +3.2%, and +8.3%, respectively. As with CTA and CPA, most methods report low performance on T2D due to limited training data.

Excluding T2D, ArcheType is the strongest baseline and outperforms Doduo on TTA: when whole table is considered as input, Doduo loses its structural advantage while ArcheType benefits from the richer semantics of an LLM over BERT. However, ArcheType still lags behind TabEmb. As a generative model, it may produce incorrect or non-canonical labels, whereas TabEmb is trained as a discriminative classifier to produce a fixed set of semantic type labels.

5.6 Analysis

We analyze TabEmb's *structure* and *semantic* components via different GNN architectures and LLM backbones, respectively; we then examine generalization to rare classes and report sensitivity to the GNN depth and the number of sampled cell values per column.

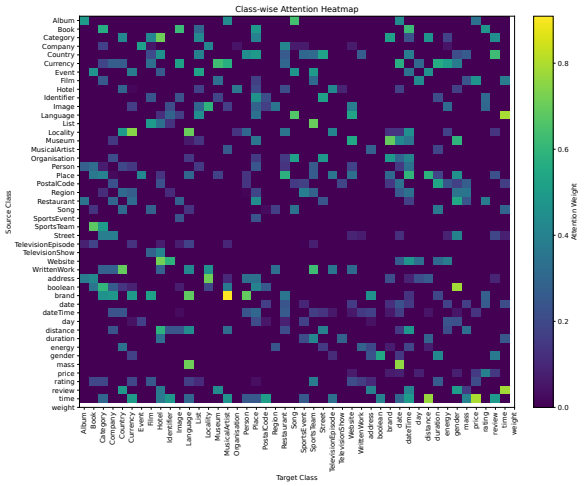


Figure 3: class–class attention heatmap on $\text{SOTAB}_{\text{dbp}}$ (CTA), aggregated from GAT attention weights across tables.

Structure Modeling. Table 7 summarizes the impact of structural modeling in TabEmb. TabEmb (no GNN) is a purely semantic variant obtained by StructEmbedding (line 15 in Algorithm 1) returning the initial GNN embeddings without any message passing, so the classifier operates directly on frozen LLM column embeddings. In contrast, TabEmb (GAT), TabEmb (GCN), and TabEmb (GGNN) enable StructEmbedding with Graph Attention Networks (GAT) (Veličković et al., 2017), Graph Convolutional Networks (GCN) (Kipf and Welling, 2016), and Gated Graph Neural Networks (GGNN) (Li et al., 2015), respectively, performing joint semantic–structure modeling over the column graph. All three structural variants improve over TabEmb (no GNN), confirming that joint semantic–structure modeling is beneficial and yields more effective embeddings. Among them, TabEmb (GAT) performs better because its attention mechanism can focus on informative columns and down-weight noisy ones, whereas GCN and GGNN aggregate neighbors more uniformly.

Figure 3 shows a class–class attention heatmap on $\text{SOTAB}_{\text{dbp}}$ (CTA), aggregated from GAT attention weights across tables, indicating that the learned column interactions are highly non-uniform. Notable interactions often appear among semantically related groups, such as location-related types (*address*, *Place*, *Street*, *PostalCode*), temporal types (*date*, *dateTime*, *time*, *duration*), and measurement-related types (*mass*, *weight*, *energy*), as well as music/media-related types (e.g., *Album*, *Song*, *MusicalArtist*).

GNN used by TabEmb	GNN Avg.
No GNN	88.3
GAT	92.1
GCN	90.5
GGNN	90.8

Table 7: Average micro-F1 scores (%) over all tasks and datasets for each GNN-based structural modeling. Per-task results are provided in Appendix B Table 12.

Semantic Modeling. Table 8 compares performance of seven $\mathcal{M}_{\text{LLM}}$ models in TabEmb. Mistral achieves the highest average score (92.1), but the improvement over other LLMs is limited, suggesting that there is not much difference between LLMs. In contrast, BERT performs worst with an overall average of 89.4, nearly 2.7 points lower than Mistral, indicating that older small PLMs like BERT capture insufficient semantic context. Since TabEmb decouples semantic encoding from graph-based structure modeling, the LLM backbone can be replaced without changing the GNN or task heads (e.g., using embedding-specialized encoders such as e5/NV-Embed/Qwen or proprietary embedding APIs).

LLM used by TabEmb	LLM Avg.
Mistral (7B)	92.1
Qwen (7B)	91.4
Mixtral (45B)	91.9
LLaMA-3 (8B)	91.3
LLaMA-2 (7B)	91.6
TinyLLaMA (1.1B)	91.3
BERT (110M)	89.4

Table 8: Average micro-F1 scores (%) over all tasks and datasets for each LLM-based semantic modeling. Per-task results are provided in Appendix C Table 13.

Generalization to Rare Classes. Table 9 reports a frequency-stratified analysis on $\text{SOTAB}_{\text{dbp}}$ for CTA/CPA/TTA. We split classes into three equally sized bins by frequency (High / Medium / Low) and report the average per-class F1 in each bin, comparing TabEmb against Doduo (the strongest joint semantic–structure baseline applicable to all three tasks). TabEmb's per-class F1 remains relatively stable across bins, indicating strong performance even on less frequent classes. Moreover, the performance gap between TabEmb and Doduo generally widens from High to Medium/Low, consistent with TabEmb's frozen LLM backbone providing richer world knowledge than Doduo's BERT encoder and thus better generalization to rare semantic types.

Sensitivity to $\mathcal{M}_{\text{GNN}}$ depth. Table 10 reports

		Frequency		
Task	**Model**	**High**	**Medium**	**Low**
CTA	TabEmb	90.2	92.8	91.2
CTA	Doduo	84.6	80.2	87.8
CPA	TabEmb	94.4	87.7	96.4
CPA	Doduo	92.0	83.1	93.2
TTA	TabEmb	97.8	95.7	92.7
TTA	Doduo	86.4	82.3	77.1

Table 9: Frequency-stratified per-class F1 on $\text{SOTAB}_{\text{dbp}}$ (CTA/CPA/TTA), comparing TabEmb vs. Doduo.

#GAT layers	1	2	3	4
Avg micro-F1	91.9	92.1	92.4	92.0

Table 10: Sensitivity to $\mathcal{M}_{\text{GNN}}$ (GAT) depth (average micro-F1 over all tasks and datasets).

TabEmb's sensitivity to $\mathcal{M}_{\text{GNN}}$ (GAT) depth, showing only modest variation across 1–4 layers, consistent with the complete-graph setting where even one layer provides global context.

Sensitivity to row number m. We observe that the number of sampled non-null cell values per column (m in the **ColumnEmbedding** function) has only a small effect once the LLM sees enough representative values: using m=25 (default) yields 92.1 avg micro-F1 vs. 91.7 for m=15, consistent with the LLM backbone providing strong semantic generalization beyond a moderate number of samples.

6 Conclusion

We presented TabEmb, a general and efficient framework for table annotation that combines frozen LLMs for semantic encoding with GNN learning for structure-aware refinement. To support the three core tasks of column type annotation, column property annotation, and table type annotation, TabEmb uses a shared LLM+GNN architecture with lightweight task-specific classifiers. This design avoids complicated prompt engineering and costly LLM fine-tuning while outperforming state-of-the-art supervised baselines across multiple benchmark datasets. Our analysis highlights how embedding design choices can impact performance, offering guidance for building effective and scalable models for structured data understanding.

Acknowledgment

The work of Ke Wang is supported in part by a discovery grant from Natural Sciences and Engineering Research Council of Canada.

Limitations

TabEmb takes longer inference time than the BERT-based Doduo (Table 3) due to incorporating large LLMs, though TabEmb improves Doduo on micro-F1 by +4.9, +5.2, +9.2 on CTA, CPA, and TTA. Therefore, Doduo remains highly competitive in the scenario where inference time outweighs accuracy, whereas TabEmb will be the choice when accuracy is crucial. In addition, TabEmb currently constructs a fully connected column graph; for very wide tables, the quadratic number of inter-column edges can make graph computation expensive. While partitioning a table's columns into several coherent blocks and constructing a fully connected graph within each block is a practical option, domain knowledge (e.g., schema constraints) can also be used to decide which column pairs should be connected by edges, introducing effective sparsity and simplifying the graph.

References

2017. T2dv2 dataset.

2023. Sotab dataset.

Jiaoyan Chen, Ernesto Jiménez-Ruiz, Ian Horrocks, and Charles Sutton. 2019a. Colnet: Embedding the semantics of web tables for column type prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 29–36.

Jiaoyan Chen, Ernesto Jiménez-Ruiz, Ian Horrocks, and Charles Sutton. 2019b. Learning semantic annotations for tabular data. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 2088–2094.

Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2020. Turl: Table understanding through representation learning. *arXiv preprint arXiv:2006.14806*.

Zhihao Ding, Yongkang Sun, and Jieming Shi. 2025. Retrieve-and-verify: A table context selection framework for accurate column annotations. *Proceedings of the ACM on Management of Data*, 3(6):1–27.

Raul Castro Fernandez, Ziawasch Abedjan, Famien Koko, Gina Yuan, Samuel Madden, and Michael Stonebraker. 2018a. Aurum: A data discovery system. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 1001–1012. IEEE.

Raul Castro Fernandez, Essam Mansour, Abdulhakim A Qahtan, Ahmed Elmagarmid, Ihab Ilyas, Samuel Madden, Mourad Ouzzani, Michael Stonebraker, and Nan Tang. 2018b. Seeping semantics: Linking datasets using word embeddings for data discovery. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, pages 989–1000. IEEE.

Benjamin Feuer, Yurong Liu, Chinmay Hegde, and Juliana Freire. 2024. Archetype: A novel framework for open-source column type annotation using large language models. *Proceedings of the VLDB Endowment*, 17(9):2279 – 2292.

Rihan Hai, Christos Koutras, Christoph Quix, and Matthias Jarke. 2023. Data lakes: A survey of functions and systems. *IEEE Transactions on Knowledge and Data Engineering*, 35(12):12571–12590.

Ehsan Hoseinzade and Ke Wang. 2024. Graph neural network approach to semantic type detection in tables. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 121–133. Springer.

Ehsan Hoseinzade and Ke Wang. 2025. Efficient table generation for zero-shot column type annotation. In *1st ICML Workshop on Foundation Models for Structured Data*.

Ehsan Hoseinzade and Ke Wang. 2026. Ztab: Domain-based zero-shot annotation for table columns. *arXiv preprint arXiv:2603.11436*.

Madelon Hulsebos, Kevin Hu, Michiel Bakker, Emanuel Zgraggen, Arvind Satyanarayan, Tim Kraska, Çagatay Demiralp, and César Hidalgo. 2019. Sherlock: A deep learning approach to semantic data type detection. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1500–1508.

Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. 2021. Tabbie: Pretrained representations of tabular data. *arXiv preprint arXiv:2105.02584*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, and 1 others. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. 2011. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3363–3372.

Moe Kayali, Anton Lykov, Ilias Fountalis, Nikolaos Vasiloglou, Dan Olteanu, and Dan Suciu. 2024. Chorus: Foundation models for unified data discovery and exploration. *Proceedings of the VLDB Endowment*, 17(8):2104–2114.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Keti Korini and Christian Bizer. 2023. Column type annotation using chatgpt. *arXiv preprint arXiv:2306.00745*.

Keti Korini and Christian Bizer. 2024. Column property annotation using large language models. *Crete, Greece*.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2015. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.

Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. 2010. Annotating and searching web tables using entities, types and relationships. *Proceedings of the VLDB Endowment*, 3(1-2):1338–1347.

Erhard Rahm and Philip A Bernstein. 2001. A survey of approaches to automatic schema matching. *the VLDB Journal*, 10(4):334–350.

Bram Steenwinckel, Gilles Vandewiele, Filip De Turck, and Femke Ongenae. 2019. Csv2kg: Transforming tabular data into semantic knowledge.

Yoshihiko Suhara, Jinfeng Li, Yuliang Li, Dan Zhang, Çağatay Demiralp, Chen Chen, and Wang-Chiew Tan. 2022. Annotating columns with pre-trained language models. In *Proceedings of the 2022 International Conference on Management of Data*, pages 1493–1503.

Yushi Sun, Hao Xin, and Lei Chen. 2023. Reca: Related tables enhanced column semantic type annotation framework. *Proceedings of the VLDB Endowment*, 16(6):1319–1331.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Daheng Wang, Prashant Shiralkar, Colin Lockard, Binxuan Huang, Xin Luna Dong, and Meng Jiang. 2021. Tcn: Table convolutional network for web table interpretation. In *Proceedings of the Web Conference 2021*, pages 4020–4032.

Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, and 1 others. 2019. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*.

Yubo Wang, Hao Xin, and Lei Chen. 2024. KGLink: A Column Type Annotation Method that Combines Knowledge Graph and Pre-Trained Language Model . In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 1023–1035, Los Alamitos, CA, USA. IEEE Computer Society.

Guorui Xiao, Dong He, Jin Wang, and Magdalena Balazinska. 2025. Cents: A flexible and cost-effective framework for llm-based table understanding. *Proceedings of the VLDB Endowment*, 18(11):4574–4587.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. *arXiv preprint arXiv:2005.08314*.

Dan Zhang, Yoshihiko Suhara, Jinfeng Li, Madelon Hulsebos, Çağatay Demiralp, and Wang-Chiew Tan. 2019. Sato: Contextual semantic type detection in tables. *arXiv preprint arXiv:1911.06311*.

Haochen Zhang, Yuyang Dong, Chuan Xiao, and Masafumi Oyamada. 2024a. Jellyfish: Instruction-tuning local large language models for data preprocessing. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8754–8782.

Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 2024b. Tablellama: Towards open large generalist models for tables. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6024–6044.

Lingfeng Zhong, Jia Wu, Qian Li, Hao Peng, and Xindong Wu. 2023. A comprehensive survey on automatic knowledge graph construction. *ACM Computing Surveys*, 56(4):1–62.

A Dataset Details

Dataset	Task	#Class	#Train	#Test
$SOTAB_{sch}$	CTA	82	44769	609
	CPA	108	29158	565
	TTA	17	29158	565
$SOTAB_{sch\text{-}s}$	CTA	82	10631	609
	CPA	108	7430	565
	TTA	17	7430	565
$SOTAB_{dbp}$	CTA	46	37631	279
	CPA	49	22905	311
	TTA	17	22905	311
T2D	CTA	37	160	109
	CPA	48	81	82
	TTA	23	81	82
Wikitable	CTA	255	397098	4764
	CPA	121	52943	1467
	TTA	N/A	N/A	N/A
Webtables	CTA	78	78733	
	CPA	N/A	N/A	N/A
	TTA	N/A	N/A	N/A

Table 11: Dataset statistics per task. For each dataset and task, we report the number of classes (#Class), training tables (#Train), and test tables (#Test). "N/A" indicates that the task is not available for the dataset.

We evaluate TabEmb on CPA, CTA and TTA tasks over six datasets. A summary of datasets used in this paper is provided in Table 11. For CTA, all six datasets are used. For CPA, all datasets (except Webtables) that provide relation-level annotations are used. For TTA, all CPA-compatible datasets except Wikitable and Webtables are repurposed. In case of T2D, we used the version

GNN used by TabEmb	Task	Task Avg.	Model Avg.
No GNN	CTA	87.50	88.29
	CPA	83.97	
	TTA	93.42	
GAT	CTA	90.67	92.09
	CPA	89.88	
	TTA	95.72	
GCN	CTA	89.59	90.48
	CPA	87.57	
	TTA	94.28	
GGNN	CTA	89.84	90.78
	CPA	88.06	
	TTA	94.45	

Table 12: Average micro-F1 scores (%) for TabEmb variants across tasks.

introduced in (Chen et al., 2019b) for CTA, and the version introduced in (Korini and Bizer, 2024) for CPA and TTA, wherein the table type labels used for the TTA task are extracted from (t2d, 2017). All three SOTAB variants define 17 distinct table types, which we use as supervision for TTA. Since Wikitable is a multi-label dataset, and TabEmb as well as several baselines require a single label per instance, we select the first available label when multiple labels are present.

B Structural Modeling

This section includes the per-task results of the structural modeling analysis under Section 5.6. Table 12 is a more detailed version of table 7.

C Semantic Modeling

This section includes the per-task results of the semantic modeling analysis under Section 5.6. Table 13 is a more detailed version of Table 8.

LLM used by TabEmb	Task	Task Avg.	LLM Avg.
Mistral (7B)	CTA	90.67	92.09
	CPA	89.88	
	TTA	95.72	
Qwen (7B)	CTA	90.52	91.40
	CPA	88.48	
	TTA	95.20	
Mixtral (45B)	CTA	91.02	91.87
	CPA	88.92	
	TTA	95.67	
LLaMA-3 (8B)	CTA	90.43	91.28
	CPA	88.61	
	TTA	94.80	
LLaMA-2 (7B)	CTA	90.49	91.63
	CPA	89.00	
	TTA	95.41	
TinyLLaMA (1.1B)	CTA	89.50	91.29
	CPA	88.72	
	TTA	95.66	
BERT (110M)	CTA	87.60	89.42
	CPA	86.93	
	TTA	93.72	

Table 13: Average micro-F1 scores (%) for each LLM and task.