Retrieve-and-Verify: A Table Context Selection Framework for Accurate Column Annotations

Zhihao Ding*
Hong Kong Polytechnic University
tommy-zh.ding@connect.polyu.hk

Yongkang Sun*
Hong Kong Polytechnic University
yongkang.sun@connect.polyu.hk

Jieming Shi
Hong Kong Polytechnic University
jieming.shi@polyu.edu.hk

Abstract

Tables are a prevalent format for structured data, yet their metadata, such as semantic types and column relationships, is often incomplete or ambiguous. Column annotation tasks, including Column Type Annotation (CTA) and Column Property Annotation (CPA), address this by leveraging table context, which are critical for data management. Existing methods typically serialize all columns in a table into pretrained language models to incorporate context, but this coarse-grained approach often degrades performance in wide tables with many irrelevant or misleading columns. To address this, we propose a novel retrieve-and-verify context selection framework for accurate column annotation, introducing two methods: REVEAL and REVEAL+. In REVEAL, we design an efficient unsupervised retrieval technique to select compact, informative column contexts by balancing semantic relevance and diversity, and develop context-aware encoding techniques with role embeddings and target-context pair training to effectively differentiate target and context columns. To further improve performance, in REVEAL+, we design a verification model that refines the selected context by directly estimating its quality for specific annotation tasks. To achieve this, we formulate a novel column context verification problem as a classification task and then develop the verification model. Moreover, in REVEAL+, we develop a top-down verification inference technique to ensure efficiency by reducing the search space for high-quality context subsets from exponential to quadratic. Extensive experiments on six benchmark datasets demonstrate that our methods consistently outperform state-of-the-art baselines.

ACM Reference Format:
Zhihao Ding, Yongkang Sun, and Jieming Shi. 2026. Retrieve-and-Verify: A Table Context Selection Framework for Accurate Column Annotations. In *Proceedings of Proceedings of the 2026 ACM SIGMOD International Conference on Management of Data (SIGMOD '26) (SIGMOD '26).* ACM, 15 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Relational tables are a fundamental format for organizing structured data in diverse applications. As structured data grows in volume and complexity, understanding tables through metadata becomes critical for efficient management and utilization [2, 39]. However, table metadata, particularly the semantic types and relationships of columns, is often missing, incomplete, or ambiguous [22, 30]. Column annotation mitigates this via important tasks, including Column Type Annotation (CTA), which predicts a target column's semantic type, and Column Property Annotation (CPA), also known as Column Relationship Annotation, which identifies relationships of column pairs. Accurate column annotation is crucial for applications such as schema matching [11, 28], dataset discovery [14, 23], data integration [24, 41], and semantic data versioning [38].

In this work, we investigate CTA and CPA in scenarios where metadata is unavailable. Accurate annotation of a target's semantic type or relationship requires the context of other columns within the same table. For example, in Table A of Figure 1(a), the target column (in gray) contains float values such as '11.11' and '13.13'. Solely relying on the target itself, it is difficult to determine if it represents PRICE or RATING semantic type. However, when considering the other columns, the target is likely to be PRICE since it is followed by a currency column, and the table describes books.

Recent studies [6, 32, 39, 40] have leveraged language models (LMs) such as BERT [8] and LLaMA [12] to enhance column annotation by utilizing column representations as context [14, 39, 47], and even inter-table columns [40]. A common approach is to serialize all columns in a table into a single sequence and input it into LMs, such as BERT in Doduo [39], to generate column embeddings

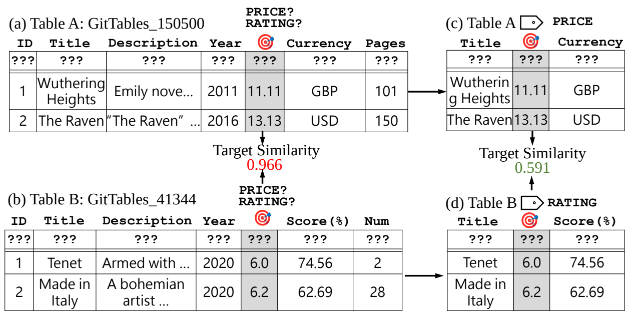

(a) Table A: GitTables_150500
PRICE?
RATING?
ID | Title | Description | Year | 🎯 | Currency | Pages
??? | ??? | ??? | ??? | ??? | ??? | ???
1 | Wuthering Heights | Emily nove... | 2011 | 11.11 | GBP | 101
2 | The Raven | "The Raven" ... | 2016 | 13.13 | USD | 150
Target Similarity
0.966
(b) Table B: GitTables_41344
PRICE?
RATING?
ID | Title | Description | Year | 🎯 | Score(%) | Num
??? | ??? | ??? | ??? | ??? | ??? | ???
1 | Tenet | Armed with ... | 2020 | 6.0 | 74.56 | 2
2 | Made in Italy | A bohemian artist ... | 2020 | 6.2 | 62.69 | 28
(c) Table A
PRICE
Title | 🎯 | Currency
??? | ??? | ???
Wuthering Heights | 11.11 | GBP
The Raven | 13.13 | USD
Target Similarity
0.591
(d) Table B
RATING
Title | 🎯 | Score(%)
??? | ??? | ???
Tenet | 6.0 | 74.56
Made in Italy | 6.2 | 62.69

Figure 1: Example on two real-world tables: (a)(c) Raw tables; (b)(d) Tables with selected contexts.

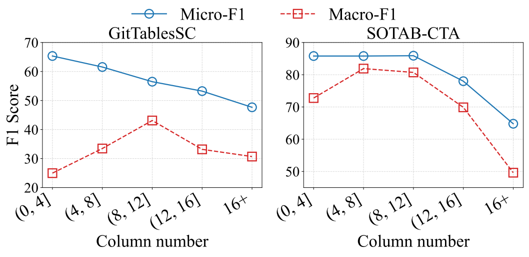


Figure 2: The impact of column number on CTA.

*Co-primary authors.

that capture patterns and relationships between columns. Then the model predicts the semantic type or relationship of the target as a classification problem based on the generated embeddings.

While existing methods have made notable progress, their approach to leveraging column context remains coarse-grained. This often leads to sub-optimal performance, particularly in wide tables with many columns, which are common in real-world scenarios [31]. We conduct an empirical analysis of Doduo on two datasets, GitTablesSC and SOTAB-CTA, to examine the impact of the number of columns in a table on model performance (see detailed experiments in Section 6). As shown in Figure 2, performance improves in terms of Macro-F1 and Micro-F1 as the number of columns increases from 1 to 12, demonstrating the benefit of leveraging additional column content. However, performance declines when the number of columns exceeds 12. This indicates that wide tables with many columns pose greater challenges, as not all columns are beneficial for the target. Including all columns may introduce noise and redundancy, degrading performance [37, 46]. We provide Example 1 with two real tables in Figure 1 to illustrate this issue.

Example 1. *Figure 1(a) and (c) show two real tables with target columns in grey and the goal is to predict their semantic types, either* `PRICE` *or* `RATING`*, which are shown for illustration but not available during annotation. Both targets contain float values, and the tables include columns with similar data types, such as text and numbers. Using all columns as context, a model may fail to distinguish the targets, leading to incorrect predictions. For instance, Doduo generates embeddings for the two targets with a high cosine similarity of 0.966, making them hard to differentiate.*

However, closer inspection reveals that the target in Table A is likely `PRICE`*, as it is followed by a currency column with GBP and USD values, while the target in Table B is likely* `RATING`*, as it is followed by a percentage column. Selecting only relevant columns, such as* `TITLE` *and* `CURRENCY` *for Table A, and* `TITLE` *and* `SCORE(%)` *for Table B, as shown in Figure 1(b) and (d), reduces the cosine similarity between the targets to 0.591, making them more distinguishable. This highlights the importance of selecting relevant context columns for accurate annotation.*

These empirical findings reveal the importance of explicitly selecting relevant column contexts for accurate column annotation. To this end, we propose a novel retrieve-and-verify context selection framework comprising the REVEAL and REVEAL+ methods. REVEAL delivers superior annotation performance with high efficiency, while REVEAL+ further enhances annotation accuracy significantly with moderate additional overhead.

In REVEAL, we first design an efficient *retrieval method* to select a compact, informative subset of columns from the input table T as the *column context* C for a given target in an unsupervised manner (Section 4.1). This method ensures that C is both semantically relevant and diverse, providing meaningful context without requiring labeled supervision. Next, to generate high-quality embeddings that effectively distinguish target and context columns, we develop *context-aware encoding techniques* (Section 4.2). This includes a context-aware encoder with role embeddings to explicitly mark column roles, and a target-context pair training strategy that treats each target-context pair as a distinct training unit, enabling effective modeling of target-context interactions.

Table 1: Frequently used notations.

Notation	Description
T	A table with multiple columns.
c_i	The i-th column in the table.
τ	The target column or column pair for annotation.
gt_τ	The ground-truth of the target.
$\mathcal{C}$	The retrieved column context for τ.
$\mathcal{S}$	The verified column context for τ.
$\mathcal{S}'$	A subset of columns from $\mathcal{C}$.
Π	The verification model.
$\Pi(\tau, \mathcal{S})$	The quality score assigned by the verification model.
f	The prediction module.
$\mathbf{h}^{\tau}_{\mathcal{S}'}$	The embedding of τ with context $\mathcal{S}'$.
$y^{\tau}_{\mathcal{S}'}$	The label indicating the quality of a column context.
K	The desired size of the retrieved column context $\mathcal{C}$.

To further improve performance, we extend REVEAL with REVEAL+, incorporating a *verification model* to refine C into a verified column context $\mathcal{S}$. The verification directly estimates the quality of the selected column context for the target in a supervised manner, ensuring that the context is not only relevant but also effective for the specific annotation tasks. To this end, we formulate a novel *column context verification* problem as a classification task, construct a pseudo-labeled dataset, and design and train the verification model (Section 5.1). To avoid the exponential search space of all subsets of C to find the highest quality $\mathcal{S}$, we develop a *top-down verification inference method* that efficiently obtains $\mathcal{S}$ using a greedy strategy, reducing the search space to quadratic (Section 5.2).

We conduct extensive experiments on six benchmark datasets, comparing our methods with state-of-the-art baselines. The results show that REVEAL and REVEAL+ consistently outperform existing approaches, demonstrating that our techniques in the proposed context selection framework advance column annotation performance.

We summarize our contributions below:

- We propose a retrieve-and-verify column context selection framework, introducing the REVEAL and REVEAL+ methods to address the challenges of noisy and redundant column contexts in column annotation tasks.
- We propose a retrieval method to select compact and informative column contexts and introduce context-aware encoding to differentiate target and context columns for better embeddings.
- We design a verification model with a novel problem formulation and an efficient top-down inference strategy to refine the selected column context, further boosting annotation performance.
- Extensive experiments on six benchmark datasets validate the effectiveness and efficiency of our proposed methods.

2 Preliminaries

We focus on two column annotation tasks: Column Type Annotation (CTA) and Column Property Annotation (CPA) [6, 32, 39, 47]. As illustrated in Figure 1, CTA assigns semantic types (e.g., `PRICE`, `RATING`) to individual columns, providing a clearer understanding of their semantics. CPA aims to determine the semantic relationship between column pairs, such as `place_of_birth` linking `person` to

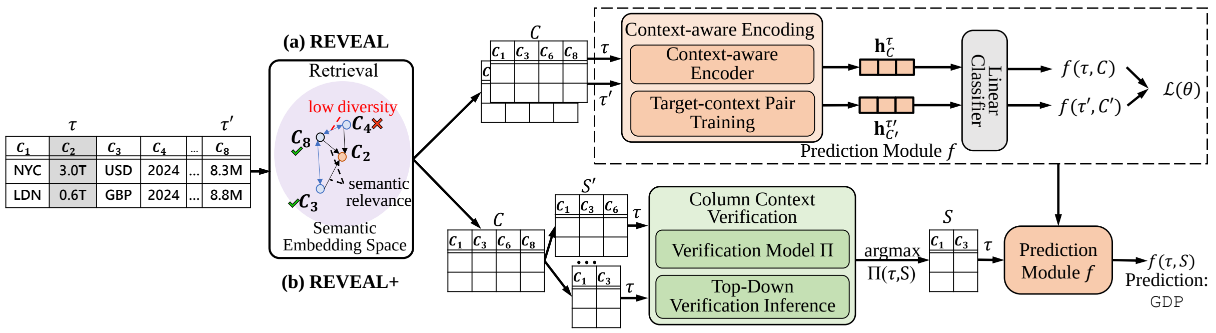


Figure 3: The Framework of REVEAL and REVEAL+.

city, or has_population linking city to population. Instead of primitive data types (e.g., String, Int, Timestamp), CTA and CPA focus on annotating columns with semantic types and relationships, offering deeper semantic understanding of tabular data. Formally, a table T with n columns is represented as $T = (c_1, c_2, \ldots, c_n)$, where c_i denotes the i-th column in the table. Each column c_i comprises m cell values, expressed as $c_i = (v_1^i, v_2^i, \ldots, v_m^i)$, where v_j^i represents the value of the j-th cell in column c_i. The definitions of CTA and CPA [32, 39] are as follows:

Definition 2 (Column Type Annotation (CTA)). *Given a table T with n columns, and a target column $\tau = c_i$, where $1 \le i \le n$, and a set of possible semantic types $\mathcal{T}$, the task of CTA is to develop a model $\mathcal{M}$ that can predict a type label $\mathcal{M}(\tau, T) \in \mathcal{T}$ so that each cell in τ has the same semantic type.*

Definition 3 (Column Property Annotation (CPA)). *Given a table T with n columns, and a target column pair $\tau = (c_i, c_j)$, where $1 \le i, j \le n$, and a set of possible relation types $\mathcal{R}$, the task of CPA is to develop a model $\mathcal{M}$ that can predict a relation type $\mathcal{M}(\tau, T) \in \mathcal{R}$ to represent the relationship between the two columns.*

We focus on the setting where no metadata, such as headers or captions, is available, as these are often missing in many datasets [18, 27]. Hereafter, when the context is clear, we use target τ to denote either a target column in CTA or a pair of target columns in CPA.

Table 1 summarizes the frequently used notations in this paper.

3 Overview

We provide an overview of our methods REVEAL and REVEAL+. Our methods are designed to be applicable to both CTA and CPA. We mainly explain our methods in the context of CTA for clarity.

Figure 3(a) illustrates the REVEAL method. Given a table T with n columns, there may be different targets, e.g., τ and τ', to be annotated. As discussed, treating all columns in T as a shared context for different targets can lead to degraded performance. Therefore, in REVEAL, we first design a retrieval method to explicitly select a subset of columns from T as the *column context* C for a target τ, considering semantic relevance and diversity among columns in T (Section 4.1). The retrieval method operates solely on unsupervised information, ensuring scalability and effectiveness. Then the next step is to encode them into embeddings via language models. Existing studies [32, 39] typically serialize all columns into a sequence, treating the target and context columns uniformly. However, we argue that the embedding of a column should adapt based on its role—whether as the target or as part of the context. To achieve this, we develop context-aware encoding techniques that explicitly differentiate the target from its context columns in embeddings (Section 4.2). Specifically, we introduce a context-aware encoder with role embeddings to mark the roles of columns during the embedding process, and we adopt a target-context pair training strategy, where each target τ is paired with its column context C as distinct training units. The obtained embeddings are then passed to a linear layer for generating the final predictions.

REVEAL excels state-of-the-art methods in effectiveness and efficiency, as shown in the experiments. While efficiency is important, annotation quality often takes precedence, especially for offline tasks. To further enhance annotation quality, REVEAL+ improves effectiveness with moderate additional computational cost.

REVEAL+ is shown in Figure 3(b). Similar to REVEAL, it starts with the same retrieval method to obtain C for the target τ. While C is selected based on unsupervised information in T, the columns in C are not yet *verified* whether they are truly helpful in the annotation tasks. To address this, REVEAL+ incorporates a verification model (Section 5) to further refine C to obtain a verified column context $\mathcal{S}$ for the target τ, which retains only the most informative columns for the target τ on the annotation tasks. To improve the efficiency of obtaining $\mathcal{S}$, we propose a top-down verification inference method (Section 5.2). As formulated in Section 5.1, the verification model Π evaluates a subset $\mathcal{S}'$ of C for a target τ by outputting a quality score $\Pi(\tau, \mathcal{S}')$, which represents the likelihood that $\mathcal{S}'$ enables the correct annotation of τ. This verification task is framed as a classification problem: determine whether a subset $\mathcal{S}'$ of C is effective for accurately annotating the target τ. The subset of C with the maximum quality score is selected as the verified column context $\mathcal{S}$ for the target τ. A key challenge is the lack of labeled data for training the verification model. To address this, we construct a labeled dataset by leveraging the predictions of the trained prediction module f on the training and validation datasets, which are available after f is trained. For each subset $\mathcal{S}' \subseteq C$, we label it as positive if f makes a correct prediction for τ using $\mathcal{S}'$ as context, indicating that $\mathcal{S}'$ is a good context. Otherwise, it is labeled as negative. The model Π is then trained on this constructed labeled dataset. During inference, the model Π finds the $\mathcal{S}$ that maximizes the score $\Pi(\tau, \mathcal{S}')$ for τ. The trained prediction module f then uses $\mathcal{S}$ to generate the final prediction for τ. Evaluating all subsets $\mathcal{S}' \subseteq C$ is infeasible due to the exponential search space of $2^{|C|}$. Hence, we propose a top-down verification inference method

(Section 5.2) that iteratively refines C, reducing the complexity to $O(|C|^2)$, with early stopping to improve efficiency.

4 The REVEAL Method

4.1 Column Context Retrieval

As mentioned, in a wide table $T = (c_1, \ldots, c_n)$, using all columns for annotating a target τ can degrade performance due to irrelevant columns. We aim to retrieve a subset of columns as the column context C for τ. However, the search space of 2^n subsets grows exponentially with n, and different targets may require distinct column contexts. In this section, we propose an efficient retrieval method that leverages only the unsupervised information of columns in T to retrieve a compact yet informative column context C for τ.

Intuitions. As shown in Figure 1, not all columns in a table equally contribute to understanding the target τ with ground truth `PRICE`. For example, the `Description` column, despite containing detailed text, is irrelevant to the target's semantics, while the `CURRENCY` column provides essential context. Including irrelevant columns can degrade performance, highlighting the need to filter for semantically relevant columns. One naive approach is to select the top-K most similar columns to τ as C. However, this may not always yield an informative context. For example, consider constructing a column context C of size 3 for the target in Figure 1(a). The target τ contains numerical values, and the cosine similarity between the target and other columns are: `ID` (0.82), `PAGES` (0.76), `YEAR` (0.68), `CURRENCY` (0.55), `TITLE` (0.28), and `DESCRIPTION` (0.21). Selecting the top-3 most similar columns would yield `ID`, `PAGES`, and `YEAR`. However, these columns are all numerically related and fail to provide diverse contextual information. In contrast, `CURRENCY` (0.55), though less similar, specifies the nature of the values in the target and is crucial for inferring its semantic type. Thus, a more balanced approach that considers both *similarity and diversity* is needed.

Retrieval Method. To select the column context C of a target τ in a table T, we consider both semantic similarity and diversity among columns. First, each column c in T is serialized into a string by concatenating its cell values $v_1, \ldots, v_m$. This serialized string is then passed through a text encoder $\mathcal{E}$ [36], which generates a dense embedding $\mathbf{e}_c$ for the column. The embeddings of semantically similar columns are closer in the embedding space.

$$\mathbf{e}_c = \mathcal{E}(\text{CONCAT}(v_1, \ldots, v_m)), \tag{1}$$

where CONCAT$(\cdot)$ represents the concatenation operation, $\mathcal{E}(\cdot)$ is the text encoder, $\mathbf{e}_c \in \mathbb{R}^{d_e}$ denotes the embedding of column $c \in T$, and d_e is the embedding dimension.

Then we use maximal marginal relevance [5] as a measure to balance relevance and diversity, and develop an iterative process to construct C. Starting with an empty $C = \emptyset$, we iteratively select the column $c \in T \setminus C$ that maximizes the marginal relevance score $g(c, \tau, C)$, and add it to C. In Equation (2), $g(c, \tau, C)$ is defined as the difference between the semantic similarity of c to the target τ, $cos(\mathbf{e}_c, \mathbf{e}_\tau)$, and the maximum similarity of c to any column already in C, $\max_{c'' \in C} cos(\mathbf{e}_c, \mathbf{e}_{c''})$. The first term ensures that the selected column is relevant to the target, while the second term penalizes redundancy by discouraging columns similar to those already in C.

Algorithm 1: Column Context Retrieval

```
Input: Table $T$, target $\tau$, encoder $\mathcal{E}$, size $K$
Output: Column context $C$
1 foreach $c \in T$ do
2     $\mathbf{e}_c \leftarrow \mathcal{E}(c)$ // Encode columns
3 $C \leftarrow \emptyset$
4 $c' \leftarrow \text{argmax}_{c \in T \setminus \{\tau\}} \cos(\mathbf{e}_c, \mathbf{e}_\tau)$ // Select the first column
5 $C \leftarrow C \cup \{c'\}$
  // Iterative selection
6 while $|C| < K$ do
7     $c' \leftarrow \text{argmax}_{c \in T \setminus C} [\cos(\mathbf{e}_c, \mathbf{e}_\tau) - \max_{c'' \in C} \cos(\mathbf{e}_c, \mathbf{e}_{c''})]$
8     $C \leftarrow C \cup \{c'\}$
9 return $C$
```

This process continues until C reaches the desired size K.

$$\begin{aligned} g(c, \tau, C) &= \cos(\mathbf{e}_c, \mathbf{e}_\tau) - \max_{c'' \in C} \cos(\mathbf{e}_c, \mathbf{e}_{c''}) \\ c' &= \operatorname*{argmax}_{c \in T \setminus C} g(c, \tau, C). \end{aligned} \tag{2}$$

Example 4. *In Figure 1(b), for a target τ containing numerical values, selecting the top-3 most similar columns retrieves other numeric columns (e.g.,* `ID`, `YEAR`, `PAGES`*), which provide limited assistance in clarifying the true semantic meaning of the target, i.e.,* `PRICE`*. In contrast, our retrieval method selects a more semantically meaningful size-3 column context C. We initialize C as empty and iteratively add columns based on their marginal relevance scores. First, the most similar column to the target,* `ID` *(0.82), is added, resulting in $C = \{\text{ID}\}$. Next, we compute $g(c, \tau, C)$ for each column $c \in T \setminus C$. For instance,* `PAGES` *has a high similarity to the target (0.76) but also a high similarity to* `ID` *(0.71), yielding $g(\text{PAGES}, \tau, C) = 0.05$. On the other hand,* `CURRENCY` *has $cos(\mathbf{e}_c, \mathbf{e}_\tau) = 0.55$ and $cos(\mathbf{e}_c, \mathbf{e}_{\text{ID}}) = 0.34$, resulting in $g(\text{CURRENCY}, \tau, C) = 0.21$, making it more favorable than* `PAGES`*. Similarly, other columns in $T \setminus C$ have lower marginal relevance scores than* `CURRENCY`*, so we expand C to $\{\text{ID}, \text{CURRENCY}\}$. Finally, we repeat the process to select the third column.* `TITLE` *achieves the highest marginal relevance score, surpassing* `PAGES`, `YEAR`*, and* `DESCRIPTION`*. Thus, the final C is $\{\text{ID}, \text{CURRENCY}, \text{TITLE}\}$, which includes* `CURRENCY` *to specify the nature of the target values, ensuring a well-rounded contextual representation.*

Algorithm 1 depicts the retrieval method. Parameter K is the desired size of column context. If a table T has columns less than K, we simply use all available columns (excluding the target) as C. Otherwise, we first compute the column embeddings for all columns in T using the text encoder $\mathcal{E}$ (Lines 1-2). We then initialize the column context C as empty and select the first column c' to be added into C as the most similar column to the target (Lines 4-5). Next, we iteratively select the next column c' from $T \setminus C$ based on its marginal relevance score $g(c, \tau, C)$, until the size of C reaches K (Lines 6-8). Specifically, the selected c' maximizes the marginal relevance score (Line 7) and is added to C (Line 8). In experiments, we have varied K to study its impact in Figure 8; the performance increases first and then remains stable after a certain K value, indicating that a moderate number of contextual columns is sufficient to provide a well-rounded context for the annotation tasks.

4.2 Context-Aware Encoding

After retrieving C for a target τ in a table T, we need to effectively leverage it for annotation. For different targets, their context columns may vary, and the model should be trained to prioritize the target while recognizing the supporting role of its context columns. To achieve this, we propose a *context-aware encoder* that incorporates column role embeddings to explicitly differentiate the target from its context columns during the embedding process. Additionally, we introduce *target-context pair training* to train the model on individual target-context pairs, ensuring that each target is paired with its specific context rather than entire tables in a single pass.

Context-Aware Encoder with Role Embeddings. Given a target τ and its column context C in a table T, we first serialize the columns in C into a token sequence by preserving their original order in the table and concatenating their cell values column by column, separated by a special token (`[CLS]` in BERT [8]) used in language models. For instance, consider the table in Figure 4: the serialization process produces the sequence "`[CLS]` NYC LDN `[CLS]` 3.0T 0.6T `[CLS]` USD GBP", where each column is delineated by a `[CLS]` token. Then for each token w_j in the sequence, the language model maps it to an embedding by aggregating a pretrained word embedding $\mathbf{w}_j$ and a position embedding $\mathbf{p}_j$, which encodes the position of the token in the sequence.

To explicitly distinguish the target from its context columns, we introduce the third embedding component for a token, *column role embedding*, in the context-aware encoder. For each token w_j in the serialized sequence, we assign a binary role indicator $r_j \in \{0, 1\}$, where $r_j = 1$ if the token belongs to the target and $r_j = 0$ otherwise. This role indicator is mapped to a role embedding vector $\mathbf{r}_j \in \mathbb{R}^d$ using a trainable lookup table $\mathbf{E}_{\text{role}} \in \mathbb{R}^{2\times d}$, where d is the embedding dimension, as follows.

$$\mathbf{r}_j = \mathbf{E}_{\text{role}}[r_j], \tag{3}$$

where $\mathbf{r}_j \in \mathbb{R}^d$ is the role embedding of token w_j.

Note that $\mathbf{E}_{\text{role}}$ contains two embeddings: one for tokens belonging to the target and another for tokens from context columns. These embeddings are jointly optimized with the other model parameters during training, enabling the model to effectively differentiate structural roles and prioritize the target.

Then, as shown in Figure 4, a token w_j is embedded into $\mathbf{x}_j$ that combines three components: the word embedding $\mathbf{w}_j$, position embedding $\mathbf{p}_j$, and role embedding $\mathbf{r}_j$ in Equation (4).

$$\mathbf{x}_j = \mathbf{w}_j + \mathbf{p}_j + \mathbf{r}_j. \tag{4}$$

Then the token embeddings $\mathbf{x}_j$ are fed into a language model, primarily composed of transformer layers with self-attention mechanisms [42], to produce the contextualized target embedding $\mathbf{h}_C^\tau$ for target τ. Specifically, $\mathbf{h}_C^\tau$ is extracted from the output corresponding to the `[CLS]` token of the target in CTA, regarding `[CLS]` as a representative token for the column, as shown in Figure 4. In CPA, with a pair of two columns as the target, we concatenate the embeddings of the columns to form the target embedding. In our design, $\mathbf{h}_C^\tau$ captures the semantics of the target, while also incorporating contextual information from the columns in C, considering their

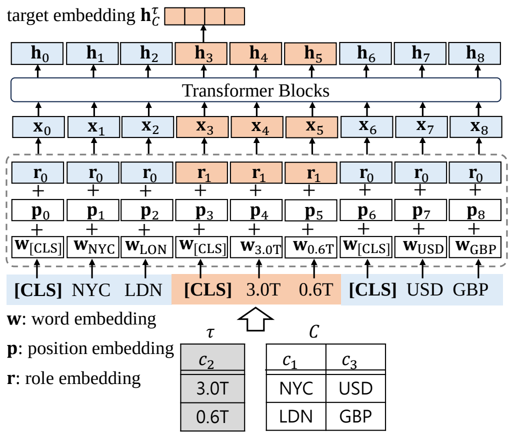

target embedding $\mathbf{h}_C^\tau$
$\mathbf{h}_0$ $\mathbf{h}_1$ $\mathbf{h}_2$ $\mathbf{h}_3$ $\mathbf{h}_4$ $\mathbf{h}_5$ $\mathbf{h}_6$ $\mathbf{h}_7$ $\mathbf{h}_8$
Transformer Blocks
$\mathbf{x}_0$ $\mathbf{x}_1$ $\mathbf{x}_2$ $\mathbf{x}_3$ $\mathbf{x}_4$ $\mathbf{x}_5$ $\mathbf{x}_6$ $\mathbf{x}_7$ $\mathbf{x}_8$
$\mathbf{r}_0$ $\mathbf{r}_0$ $\mathbf{r}_0$ $\mathbf{r}_1$ $\mathbf{r}_1$ $\mathbf{r}_1$ $\mathbf{r}_0$ $\mathbf{r}_0$ $\mathbf{r}_0$
$\mathbf{p}_0$ $\mathbf{p}_1$ $\mathbf{p}_2$ $\mathbf{p}_3$ $\mathbf{p}_4$ $\mathbf{p}_5$ $\mathbf{p}_6$ $\mathbf{p}_7$ $\mathbf{p}_8$
$\mathbf{w}_{\text{[CLS]}}$ $\mathbf{w}_{\text{NYC}}$ $\mathbf{w}_{\text{LON}}$ $\mathbf{w}_{\text{[CLS]}}$ $\mathbf{w}_{3.0\text{T}}$ $\mathbf{w}_{0.6\text{T}}$ $\mathbf{w}_{\text{[CLS]}}$ $\mathbf{w}_{\text{USD}}$ $\mathbf{w}_{\text{GBP}}$
[CLS] NYC LDN [CLS] 3.0T 0.6T [CLS] USD GBP
w: word embedding
p: position embedding
r: role embedding
τ
C
c_2
3.0T
0.6T
c_1 c_3
NYC USD
LDN GBP

Figure 4: Illustration of Context-Aware Column Embedding.

distinct roles. Then a linear layer classifier ψ is applied to $\mathbf{h}_C^\tau$ to predict the semantic type type or relation of the target,

$$f(\tau, C) = \psi(\mathbf{h}_C^\tau), \tag{5}$$

where the prediction module $f(\tau, C)$ denotes the predictions for the target τ based on the given context C.

Target-Context Pair Training. Prior works [14, 32, 39] typically use the entire table T as the training instance for different targets, resulting in all targets sharing the same table context. This approach may include columns that are irrelevant or uninformative for specific targets, potentially degrading model performance. In contrast, our design allows different targets in the same table T to have distinct column contexts C, tailored to each target. After obtaining the column context C for a target τ in a table T, we pair the target τ with its unique context columns C and the corresponding ground-truth label gt_τ, forming a training instance (τ, C, gt_τ). This reformulates the training data into a set of target-context pairs, enabling the model to focus on target-specific contexts. Since the retrieval method in Section 4.1 is unsupervised, this process is efficient and does not require additional supervision. Formally, for each labeled target in the original training set $\mathcal{D}^{\text{train}}$, we construct the target-context pair training set $\mathcal{D}^{\text{train}}_{\text{pair}}$ as follows:

$$\mathcal{D}^{\text{train}}_{\text{pair}} = \{(\tau, C, gt_\tau) \mid \tau \in \mathcal{D}^{\text{train}}\}, \tag{6}$$

where gt_τ is the ground truth of τ.

Our method REVEAL is then trained using this target-context pair training set, optimizing the following objective:

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{D}^{\text{train}}_{\text{pair}}|} \sum_{(\tau, C, gt_\tau)\in\mathcal{D}^{\text{train}}_{\text{pair}}} \ell(f(\tau, C), gt_\tau) \tag{7}$$

where $f(\tau, C)$ is predication for target τ, and ℓ is the cross-entrpoy loss, and θ represents the trainable parameters.

5 The REVEAL+ Method

Given a target τ in a table T, we efficiently obtain its column context C in Section 4.1 using unsupervised information, which is independent of specific CTA/CPA tasks. However, the retrieved context columns in C are *not yet verified* for their effectiveness in the specific CTA/CPA tasks. In this section, we propose REVEAL+, which incorporates a *verification method* to refine C by directly evaluating the effectiveness of its columns for the target τ for the annotation tasks. This refinement produces a *verified column context* $\mathcal{S}$, which improves the annotation performance of REVEAL+ compared to REVEAL. We first formulate the column verification problem as a classification task and present the verification model design (Section 5.1). To further enhance efficiency, we propose a top-down verification inference technique to greedily identify $\mathcal{S}$ from C without exhaustively evaluating all subsets (Section 5.2).

5.1 Verification Model Formulation and Design

During inference, given a table T, we aim to identify a verified column context $\mathcal{S} \subseteq C$ that is most effective for the target τ. A naive approach is to exhaustively evaluate all possible subsets of C using the trained prediction module component f (as shown in Figure 3(a)) and select the subset with the highest prediction confidence as $\mathcal{S}$. However, prior research has demonstrated that model confidence is often poorly calibrated and does not reliably indicate prediction correctness [16], result in suboptimal or misleading decisions, as validated by experiments in Section 6.4. Moreover, exhausting all subsets is computationally expensive.

On the other hand, we leverage the training and validation datasets with ground truth to train a lightweight verification model Π, which produces a quality score $\Pi(\tau, \mathcal{S}')$ to evaluate the quality of a subset $\mathcal{S}' \subseteq C$ as a verified column context for the target τ. Below, we formulate column context verification as a classification task. The formal column context verification problem is to find $\mathcal{S}$ that maximizes the quality score $\Pi(\tau, \mathcal{S}')$, as follows:

Definition 5 (Column Context Verification). *Given a table T with a target τ and its column context C, let $\mathbb{S} = \{\mathcal{S}' \mid \mathcal{S}' \subseteq C\}$. Column context verification aims to identify the subset $\mathcal{S} \in \mathbb{S}$ that maximizes the quality score, i.e.,*

$$\mathcal{S} = \operatorname*{argmax}_{\mathcal{S}' \in \mathbb{S}} \Pi(\tau, \mathcal{S}'), \tag{8}$$

where Π evaluates the effectiveness of $\mathcal{S}'$ for annotating τ.

A key challenge in implementing the verification model Π is the lack of labeled training data. There are no predefined labels indicating whether a subset of columns $\mathcal{S}'$ is effective for annotating a target τ. Manually defining rules for labeling may not align with actual annotation performance and fails to account for the target-specific nature of context quality. Different targets within the same table often require distinct context columns, making the labeling process inherently complex and context-dependent. To address this, we propose to construct labeled data for Π using the training and validation datasets, as these datasets already contain ground-truth labels. This approach ensures that the verification model Π is aligned with the trained prediction module f in Figure 3(a), as Π is trained using the predictions of f on these datasets.

Labeled Data for Verification Model. Given a training or validation sample with a target τ in a table T and its retrieved column context C, we determine whether a subset $\mathcal{S}' \subseteq C$ enables the trained prediction module f to make a correct prediction for τ. Specifically, for each subset $\mathcal{S}'$, we obtain the prediction $f(\tau, \mathcal{S}')$ via Equation (5), and compare it with the ground-truth label gt_τ. This binary outcome (correct or incorrect prediction) serves as the label $y^\tau_{\mathcal{S}'}$ for training our verification model. Formally, as defined in Equation (9), if the prediction $f(\tau, \mathcal{S}')$ matches the ground truth gt_τ, we label $y^\tau_{\mathcal{S}'} = 1$ (positive, i.e., high quality); otherwise, we label $y^\tau_{\mathcal{S}'} = 0$ (negative, i.e., low quality).

$$y^\tau_{\mathcal{S}'} = \begin{cases} 1, & \text{if } f(\tau, \mathcal{S}') = gt_\tau, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

To create the labeled data $\mathcal{D}^{\text{train}}_{\text{v}}$ for the verification model, for each training and validation table T with target τ, ground truth gt_τ, and column context C, we evaluate every subset $\mathcal{S}' \subseteq C$. Using f, we compute the prediction $f(\tau, \mathcal{S}')$ and assign the label $y^\tau_{\mathcal{S}'}$ for $\mathcal{S}'$ based on Equation (9), resulting in a labeled sample $(\tau, \mathcal{S}', y^\tau_{\mathcal{S}'})$.

Both training and validation datasets are used to construct $\mathcal{D}^{\text{train}}_{\text{v}}$ for the following reasons. The prediction module f, trained on the training dataset, tends to produce mostly positive labels ($y^\tau_{\mathcal{S}'} = 1$), which may lead to insufficient negative samples. In contrast, the validation dataset, unseen during f training, provides more negative examples ($y^\tau_{\mathcal{S}'} = 0$), ensuring a balanced dataset for training Π.

The construction of $\mathcal{D}^{\text{train}}_{\text{v}}$ occurs after f is trained, ensuring no interference between their training processes. During inference, the trained Π is used to identify the verified column context $\mathcal{S}$, which enhances the predictions of the trained f, as shown in Figure 3(b).

Verification Model Design. The architecture of the verification model Π is designed to predict the label $y^\tau_{\mathcal{S}'}$ (0 or 1) for a given target τ in table T and a subset of columns $\mathcal{S}'$. A simple implementation could treat Π as a binary classifier, outputting a single logit transformed by a sigmoid function to estimate $\Pi(\tau, \mathcal{S}')$, the probability that f correctly predicts τ's label given context $\mathcal{S}'$. However, as noted in [17], such an approach may lead to overconfident positive predictions, as it does not explicitly model the likelihood of negative outcomes.

To address this, we implement Π as a lightweight multilayer perceptron (MLP) with an output layer producing two logits: one for the positive class and one for the negative class. The softmax probability of the positive class is used as the quality score. This design ensures that when the verification model is uncertain about a context, the logits for both classes are close, resulting in a softmax score near 0.5. Conversely, when the model is confident, the logits are skewed, yielding a score closer to 0 or 1. We use the context-aware encoding technique from Section 4.2 to compute the embedding $\mathbf{h}^\tau_{\mathcal{S}'}$ for the target τ given the subset $\mathcal{S}'$. The MLP in the verification model then processes $\mathbf{h}^\tau_{\mathcal{S}'}$ and outputs a two-dimensional logit vector $\mathbf{z}^\tau_{\mathcal{S}'} \in \mathbb{R}^2$, corresponding to the logits for the negative and positive classes:

$$\mathbf{z}^\tau_{\mathcal{S}'} = \text{MLP}(\mathbf{h}^\tau_{\mathcal{S}'}). \tag{10}$$

The predicted quality score is then computed as the softmax probability of the positive class:

$$\Pi(\tau, \mathcal{S}') = \frac{e^{z^\tau_{\mathcal{S}',1}}}{\sum_{j=0}^{1} e^{z^\tau_{\mathcal{S}',j}}}. \tag{11}$$

The loss function $\mathcal{L}_{\text{v}}$ for training the verification model is

$$\mathcal{L}_{\text{v}}(\theta_\Pi) = -\sum_{(\tau, \mathcal{S}', y^\tau_{\mathcal{S}'}) \in \mathcal{D}^{\text{train}}_{\text{v}}} \left[(1 - y^\tau_{\mathcal{S}'}) \log(1 - \Pi(\tau, \mathcal{S}')) + y^\tau_{\mathcal{S}'} \log \Pi(\tau, \mathcal{S}') \right], \tag{12}$$

where θ_Π represents the trainable parameters.

5.2 Top-Down Verification Inference

After training the verification model Π in Section 5.1, we use it during inference to identify the verified column context $\mathcal{S}$ from C for the target τ. However, evaluating all $2^{|C|}$ possible subsets of C is computationally prohibitive. Instead, we propose a top-down inference method to efficiently search for $\mathcal{S}$ in a greedy manner.

The method starts with the full C and iteratively removes columns that are less informative to the target τ. This approach leverages the fact that the retrieved context columns in C from Section 4.1 are already high-quality, with only a few potentially noisy or irrelevant ones. Larger contexts generally provide richer semantic information, offering a more comprehensive understanding of the target. However, excessively small subsets may amplify the influence of individual misleading columns, degrading predictions. The top-down verificaiton process makes greedy decisions and incorporates early stopping to balance efficiency and effectiveness.

Specifically, let $\mathcal{S}^{(t)}$ denote the verified column context obtained in the t-th iteration. The top-down inference starts from $\mathcal{S}^{(t=0)} = C$ of size $|C|$ for the target τ in a table T. In the next $(t+1)$ iteration, we generate all subsets of $\mathcal{S}^{(t)}$ by removing a single column from it, i.e., $\mathcal{S}' \subset \mathcal{S}^{(t)}$ with $|\mathcal{S}'| = |\mathcal{S}^{(t)}| - 1$. For all these subsets, we compute their quality scores $\Pi(\tau, \mathcal{S}')$ by Equations (10) and (11). We then select the subset $\mathcal{S}'$ with the highest quality score as the new verified column context $\mathcal{S}^{(t+1)}$, to be used in the next iteration. The computation is formally described as:

$$\mathcal{S}^{(t+1)} = \underset{\mathcal{S}' \subset \mathcal{S}^{(t)},\, |\mathcal{S}'| = |\mathcal{S}^{(t)}| - 1}{\operatorname{argmax}} \Pi(\tau, \mathcal{S}'), \tag{13}$$

This greedy refinement process continues until the size of $\mathcal{S}^{(t)}$ reaches 1 or the following early stop condition is met: If the new verified column context $\mathcal{S}^{(t+1)}$ has a lower quality score than the previous one $\mathcal{S}^{(t)}$, meaning that unlikely the new $\mathcal{S}^{(t+1)}$ can help to make better predictions for τ, and the subsequent iterations may not yield better results. In this case, we stop the refinement process and keep the current $\mathcal{S}^{(t)}$ as the final verified column context $\mathcal{S}$ for the input target τ. Formally, we check the following early-stop condition:

$$\Pi(\tau, \mathcal{S}^{(t+1)}) < \Pi(\tau, \mathcal{S}^{(t)}). \tag{14}$$

This top-down inference technique reduces the search space from $2^{|C|}$ to at most $|C|^2$ (in the worst case). In practice, it is much smaller due to early stopping.

Algorithm. Algorithm 2 summarizes the top-down inference process. In Lines 1-3, we initialize the verified column context $\mathcal{S}^{(0)}$ as C and compute its quality score using the verification model Π with Equations (10) and (11). In Lines 4-15, we iteratively refine the verified column context $\mathcal{S}^{(t)}$ by generating all subsets of size $|\mathcal{S}^{(t)}| - 1$ and selecting the one with the highest quality score. The while loop terminates when the size of $\mathcal{S}^{(t)}$ reaches 1 or the early stop condition is met at Lines 13-14. In the t-th iteration, we generate all subsets of size $|\mathcal{S}^{(t)}| - 1$ (Line 5), and initialize the best subset $\mathcal{S}^{(t+1)}$ as empty and its quality score as $-\infty$ (Line 6-7). Then we iterate over all subsets $\mathcal{S}'$ in $\mathbb{S}$ (Line 8) and compute their quality scores using the verification model Π (Line 9). If the current subset $\mathcal{S}'$ has higher $\Pi(\tau, \mathcal{S}')$ than the best subset $\mathcal{S}^{(t+1)}$ so far, we update the best subset $\mathcal{S}^{(t+1)}$ and its quality score to be the current subset

Algorithm 2: Top-Down Inference for Verified Column Context

```
Input: target $\tau$, column context $C$, trained verification model $\Pi$
Output: Verified column context $\mathcal{S}$
1  $t \leftarrow 0$;
2  $\mathcal{S}^{(t)} \leftarrow C$;
3  Get $\Pi(\tau, \mathcal{S}^{(t)})$ by Equations (10) and (11);
4  while $|\mathcal{S}^{(t)}| > 1$ do
5      $\mathbb{S} \leftarrow \{\mathcal{S}' \subset \mathcal{S}^{(t)} \mid |\mathcal{S}'| = |\mathcal{S}^{(t)}| - 1\}$;
6      $\mathcal{S}^{(t+1)} \leftarrow \emptyset$;
7      $\Pi(\tau, \mathcal{S}^{(t+1)}) \leftarrow -\infty$;
8      foreach $\mathcal{S}' \in \mathbb{S}$ do
9          Get $\Pi(\tau, \mathcal{S}')$ by Equations (10) and (11);
10         if $\Pi(\tau, \mathcal{S}') > \Pi(\tau, \mathcal{S}^{(t+1)})$ then
11             $\mathcal{S}^{(t+1)} \leftarrow \mathcal{S}'$; // Update best subset
12             $\Pi(\tau, \mathcal{S}^{(t+1)}) \leftarrow \Pi(\tau, \mathcal{S}')$; // Update best score
13     if $\Pi(\tau, \mathcal{S}^{(t+1)}) < \Pi(\tau, \mathcal{S}^{(t)})$ then
14         break; // Early stopping
15     $t \leftarrow t + 1$;
16 $\mathcal{S} \leftarrow \mathcal{S}^{(t)}$;
17 return $\mathcal{S}$;
```

$\mathcal{S}'$ and its quality score $\Pi(\tau, \mathcal{S}')$ (Line 10-12). Lines 13-14 check the early stop condition. If the new verified column context $\mathcal{S}^{(t+1)}$ has a lower quality score than the previous one $\mathcal{S}^{(t)}$, we stop the process. Otherwise, we update the iteration index t and continue to the next iteration (Line 15). Finally, we return the verified column context $\mathcal{S}$ as the final output (Lines 16-17).

5.3 Training Process

We summarize the training process of REVEAL+ in Algorithm 3. Given a training set $\mathcal{D}^{\text{train}}$, we obtain the column contexts C for the targets using the retrieval algorithm in Section 4.1 (Line 1). Then we obtain the target-context pair training set $\mathcal{D}^{\text{train}}_{\text{pair}}$ using Equation (6) (Line 2). As introduced in Section 4.2, this dataset includes pairs of target columns (or column pairs for CPA) and their retrieved context columns. Then, we train the prediction module f using $\mathcal{D}^{\text{train}}_{\text{pair}}$ and the loss function $\mathcal{L}(\theta)$ defined in Equation (7) (Line 3). Once the prediction module f is trained, we construct the verification training set $\mathcal{D}^{\text{train}}_{\text{v}}$ from $\mathcal{D}^{\text{train}}_{\text{pair}}$ as described in Equation (9) (Line 4). This set consists of labeled context subsets based on whether they enable correct predictions by the trained prediction module. Finally, in Line 5, we train the verification model Π using $\mathcal{D}^{\text{train}}_{\text{v}}$ with the binary cross-entropy loss $\mathcal{L}_{\text{v}}(\theta_{\Pi})$ defined in Equation (12). The procedure concludes by returning the trained f and Π (Line 6). When Lines 4 and 5 are skipped, it is the training process of REVEAL, which only involves context retrieval and training the prediction module f.

The inference processes of REVEAL and REVEAL+ have already been illustrated in Figure 3.

6 Experiments

We conduct extensive experiments to evaluate our proposed methods, REVEAL and REVEAL+, on real-world datasets for both CTA and CPA tasks. The implementation is publicly available at: https://github.com/TommyDzh/REVEAL.

Algorithm 3: REVEAL+ Training Procedure

```
Input: training set $\mathcal{D}^{\text{train}}$, initialized prediction module $f$ and
       verification model $\Pi$
Output: trained prediction model $f$, verification model $\Pi$
1 Obtain column contexts $C$ for training targets in $\mathcal{D}^{\text{train}}$ by
    Algorithm 1;
2 Construct $\mathcal{D}^{\text{train}}_{\text{pair}}$ from $\mathcal{D}^{\text{train}}$ by Equation (6);
3 Train $f$ on $\mathcal{D}^{\text{train}}_{\text{pair}}$ using $\mathcal{L}(\theta)$ in Equation (7);
4 Construct $\mathcal{D}^{\text{train}}_{\text{v}}$ from $\mathcal{D}^{\text{train}}_{\text{pair}}$ by Equation (9);
5 Train $\Pi$ on $\mathcal{D}^{\text{train}}_{\text{v}}$ using $\mathcal{L}_{\text{v}}(\theta_{\Pi})$ in Equation (12);
6 return trained $f$, $\Pi$;
```

Table 2: Dataset statistics.

Benchmark	# Tables	# Types	Total # Cols	# Labeled Cols	Min/Max/Avg Cols per Table
GitTablesDB	3,737	101	45,304	5,433	1 / 193 / 12.1
GitTablesSC	2,853	53	34,148	3,863	1 / 150 / 12.0
SOTAB-CTA	24,275	91	195,543	64,884	3 / 30 / 8.1
SOTAB-CPA	20,686	176	196,831	74,216	3 / 31 / 9.5
WikiTables-CTA	406,706	255	2,393,027	654,670	1 / 99 / 5.9
WikiTables-CPA	55,970	121	306,265	62,954	2 / 38 / 5.5

6.1 Experiment Setup

Datasets. We use six benchmark datasets with real-world tables, four for CTA (GitTablesDB, GitTablesSC, SOTAB-CTA, WikiTables-CTA) and two for CPA (SOTAB-CPA, WikiTables-CPA). Table 2 summarizes the dataset statistics. GitTablesDB and GitTablesSC are part of the SemTab 2022 benchmark [1], sourced form GitTables [18], and are both used for the CTA task. GitTablesDB columns are annotated with DBpedia properties, while GitTablesSC uses Schema.org properties. The tables in the two datasets are wide. For example, the average of columns per table in GitTablesDB is 12.1, with some tables having up to 193 columns. The SOTAB-CTA and SOTAB-CPA datasets are part of the WDC Schema.org Table Annotation Benchmark (SOTAB) [27], which targets the CTA and CPA tasks using 91 Schema.org types and 176 Schema.org relations, respectively. The number of columns in the tables in these datasets can be up to 31. The WikiTables-CTA and WikiTables-CPA datasets are based on the WikiTables corpus from Wikipedia and introduced by TURL [6]. They are used for CTA and CPA, respectively, with 255 DBpedia types and 121 DBpedia relations. Note that in these datasets, we uses all columns from the original tables. The average number of columns per table in WikiTables-CTA and WikiTables-CPA is not large, but the tables can still be wide with the average number of columns being 5.9 and 5.5, respectively.

Baselines. We compare REVEAL and REVEAL+ against strong baselines, including six state-of-the-art models specifically designed for annotation tasks and two large language models (LLMs).

- **TURL** [6] adapts the transformer architecture to better capture table structure. It is pretrained on a large-scale table corpus.
- **Sato** [47] uses table-level features and models pairwise dependencies between neighboring columns for column type predictions.
- **Doduo** [39] introduces a serialization strategy for tables, enabling language models to effectively encode columns for annotation tasks. It is fine-tuned via multi-task training.
- **RECA** [40] utilizes named entity-based schema alignment to consider inter-table information.
- **Starmie** [14] develops a contrastive multi-column pretraining to enhance embeddings by incorporating semantics within tables.
- **Watchog** [32] leverages contrastive learning on unlabeled table corpora to learn representations for table understanding tasks.
- **TableLlama** [48] is an open-source generalist model for a various table-based tasks. It fine-tunes LLMs on an extensive corpus of tables and associated tasks.
- **Qwen-Plus (0-shot / 5-shot)** [44] is an LLM with strong capabilities on NLP and table-based tasks. We evaluate both 0-shot and 5-shot in-context learning variants, following the setup in [25, 26].

Evaluation Metrics. For effectiveness, we adopt the standard evaluation metrics used in prior work [6, 18, 27] for CTA and CPA tasks, including Micro-F1 and Macro-F1, since the tasks are multi-class classification problems. Following prior work, we evaluate performance using ground-truth labels, without considering semantic hierarchies or specificity. A prediction is correct if it matches the ground truth. Micro-F1 is the weighted average of F1 scores of classes, where each class contributes proportionally to its number of samples. It reflects overall performance but is biased toward frequent classes. Macro-F1 is the unweighted average of F1 scores across all classes, treating each class equally. It is more sensitive to performance on rare or underrepresented classes. We perform 5-fold cross-validation on GitTablesDB and GitTablesSC following [32]. For the remaining datasets, we train and evaluate each model five times with different random seeds, except the LLM baselines that directly use task-specific prompts for predictions [48] as explained below. We report the mean and standard deviation of results over five runs.

Implementations. For baselines, we obtain their publicly available codebases and adopt them for the CTA and CPA tasks. We adopt BERT as the base language model for Doduo, Starmie, RECA, TURL and Watchog to ensure a fair comparison. For LLM-based methods TableLlama and Qwen-Plus, we utilize task-specific prompts for CTA and CPA, following the prompt templates provided in [48], to generate predictions for column types and relations. We run the offically released TableLlama and access Qwen-Plus API for 0-shot and 5-shot in-context predictions. For 5-shot, demonstrations are randomly sampled from the training set for each test instance, following [25, 26]. We implement our methods REVEAL and REVEAL+ in Python using PyTorch and the Transformer library [43]. The context-aware encoder is built on BERT as well. We use the Adam optimizer to train. The learning rate is set to 5e-5, the desired size of column context K is fixed at 8 across all datasets, and the maximum input sequence length is set to 256 tokens. For long columns, we follow prior work [32, 39] by allocating tokens equally across columns and truncating row-wise, using the earliest rows that fit within the token limit. We select the best model checkpoint based on the highest Micro-F1 or Macro-F1 score on validation data. For the verification model in REVEAL+, it is a three-layer MLP trained after the annotation model. We use a batch size of 64 for GitTablesDB and GitTablesSC, and 512 for the other datasets. The learning rate is tuned from $\{5e{-}5, 1e{-}4, 5e{-}4\}$. All experiments are conducted on a Linux server with an Intel Xeon Gold 6226R CPU with 2.90GHz and an NVIDIA RTX 3090 GPU.

Table 3: Overall performance by Micro-F1 (%) and Macro-F1 (%) (mean ± std).

Method	GitTablesDB		GitTablesSC		SOTAB-CTA		SOTAB-CPA		WikiTables-CTA		WikiTables-CPA		Avg. Rank
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	
TURL	48.20 ± 0.98	19.56 ± 2.45	58.15 ± 0.98	26.26 ± 4.70	80.13 ± 0.25	77.34 ± 0.98	66.11 ± 0.31	60.52 ± 0.27	90.53 ± 0.07	66.40 ± 0.23	89.60 ± 0.03	80.49 ± 0.36	7.42
Sato	46.05 ± 0.87	23.50 ± 1.01	55.83 ± 0.34	30.07 ± 0.94	71.74 ± 0.25	70.01 ± 0.27	54.06 ± 0.15	46.59 ± 0.24	61.56 ± 0.84	32.05 ± 0.72	77.09 ± 0.15	49.70 ± 0.26	8.33
Doduo	44.77 ± 4.36	21.63 ± 3.26	52.36 ± 2.94	25.09 ± 5.01	81.32 ± 0.00	78.51 ± 0.00	78.85 ± 0.49	74.95 ± 0.49	92.19 ± 1.01	74.10 ± 0.86	91.60 ± 0.23	83.81 ± 0.15	6.58
Starmie	54.43 ± 0.93	30.71 ± 3.58	64.87 ± 2.47	37.04 ± 4.33	87.95 ± 0.27	86.84 ± 0.08	78.92 ± 0.13	75.69 ± 0.31	92.91 ± 0.08	76.45 ± 0.44	92.51 ± 0.36	85.69 ± 0.47	3.50
RECA	53.83 ± 1.67	25.70 ± 5.49	64.52 ± 1.69	35.10 ± 3.81	68.52 ± 0.35	68.21 ± 0.71	54.31 ± 0.99	47.56 ± 0.95	91.27 ± 0.03	73.36 ± 0.82	88.90 ± 0.94	78.39 ± 1.09	7.17
Watchog	53.96 ± 0.78	28.89 ± 3.34	65.24 ± 1.90	36.06 ± 3.39	86.23 ± 0.22	84.19 ± 0.06	76.52 ± 0.11	72.60 ± 0.25	92.25 ± 0.06	73.77 ± 0.35	92.28 ± 0.00	85.27 ± 0.25	4.75
Qwen-Plus (0-shot)	53.51 ± 0.60	27.75 ± 1.81	66.80 ± 1.82	40.35 ± 2.43	48.26 ± 0.00	44.36 ± 0.00	52.68 ± 0.00	44.01 ± 0.00	31.70 ± 0.00	12.28 ± 0.00	27.63 ± 0.00	18.13 ± 0.00	8.67
Qwen-Plus (5-shot)	57.48 ± 1.20	33.66 ± 1.57	63.21 ± 1.24	44.81 ± 2.60	52.61 ± 0.00	48.82 ± 0.00	53.48 ± 0.00	45.41 ± 0.00	32.21 ± 0.00	13.01 ± 0.00	28.18 ± 0.00	18.35 ± 0.00	7.58
TableLlama	11.03 ± 0.74	9.70 ± 1.53	12.85 ± 0.91	11.71 ± 1.62	23.35 ± 0.00	20.59 ± 0.00	16.96 ± 0.00	13.81 ± 0.00	91.85 ± 0.00	76.04 ± 0.00	92.12 ± 0.00	85.67 ± 0.00	8.92
REVEAL	59.79 ± 0.98	36.40 ± 3.17	69.26 ± 0.93	41.98 ± 1.96	88.35 ± 0.00	87.60 ± 0.00	80.45 ± 0.25	77.81 ± 0.24	**93.14 ± 0.51**	77.81 ± 0.43	92.76 ± 0.11	86.40 ± 0.07	2.00
REVEAL+	**61.53 ± 2.27**	**38.30 ± 1.16**	**70.90 ± 2.18**	**45.82 ± 1.16**	**88.74 ± 0.22**	**88.10 ± 0.06**	**80.81 ± 0.07**	**78.15 ± 0.15**	93.00 ± 0.04	**78.07 ± 0.22**	**92.80 ± 0.18**	**86.81 ± 0.23**	**1.08**

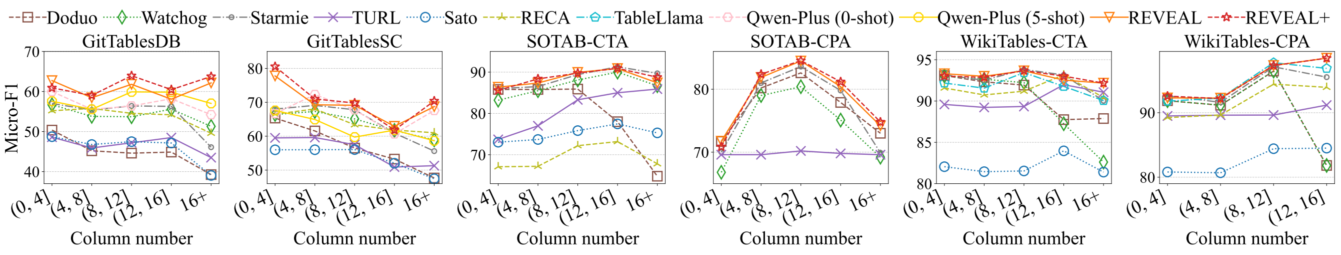


Figure 5: Performance on tables with a small to large number of columns.

6.2 Overall Results

Table 3 reports the overall performance of REVEAL and REVEAL+ compared to all baselines across all benchmark datasets. The best, second, and third performing methods are highlighted in bold, underlined, and double-underlined, respectively.

The overall observation is that REVEAL and REVEAL+ consistently outperform all baselines across all datasets under both metrics, demonstrating the effectiveness of our proposed techniques for CTA and CPA tasks. REVEAL+ and REVEAL achieve the average rank of 1.08 and 2.0, respectively, indicating their superior performance compared to the other methods. Especially on the datasets with wide tables with large average columns per table (GitTablesDB, GitTablesSC, SOTAB-CTA, and SOTAB-CPA), REVEAL and REVEAL+ achieve significant improvements over the best baselines in Micro-F1 and Macro-F1 metrics. For example, on GitTablesDB, REVEAL+ achieves 4.05% and 4.64% improvement in Micro-F1 and Macro-F1 over the best baseline Qwen-Plus (5-shot), and REVEAL also achieves significant improvements of 2.31% and 2.74% in Micro-F1 and Macro-F1. On SOTAB-CPA, REVEAL+ and REVEAL achieve significant improvements as well. REVEAL+ improves Macro-F1 by 2.46% and REVEAL by 2.12% over the best baseline Starmie. On WikiTables-CTA and WikiTables-CPA, where the average number of columns per table is not large, baselines perform well, but REVEAL and REVEAL+ still outperform all baselines. The overall results demonstrate the effectiveness of our proposed retrieval, context-aware encoding, and verification techniques in Sections 4.1, 4.2 and 5, to select high-quality context columns for accurate CTA and CPA tasks, especially in handling wide tables.

Moreover, observe that REVEAL+ consistently outperforms REVEAL across almost all datasets under Micro-F1 and Macro-F1, except WikiTables-CTA where REVEAL performs slightly better in Micro-F1. The average rank of REVEAL+ is 1.08. The improvement of REVEAL+ over REVEAL is particularly significant in Macro-F1, which treats majority and minority classes equally. For example, on GitTablesDB and GitTablesSC, REVEAL+ improves Macro-F1 by 1.90% and 3.84%, respectively. This indicates the verification model in Section 5 is effective in refining the retrieved context columns and improving the performance.

Besides, among the baselines, Starmie performs the best, with average rank of 3.5, and it adopts contrastive pretraining to capture column semantics. The baselines designed for annotation tasks, e.g., Watchog and Doduo, typically achieve better performance than the LLM-based methods. This indicates that specific techniques need to be designed for table annotation tasks. The LLMs yield unstable performance across datasets. Qwen-Plus (0-shot) and (5-shot) perform well on GitTablesDB and GitTablesSC, but they are significantly worse on the other four datasets, with average ranks of 8.67 and 7.58, respectively; TableLlama achieves strong results in WikiTables-CTA and WikiTables-CPA but fails in the other four datasets, resulting in rank of 8.92. This inconsistency may depend on if similar table corpus were seen during its pretraining or not. These results suggest that table annotation tasks remain challenging and cannot be totally solved by LLMs alone.

Moreover, we report that REVEAL+ obtains a subset of columns for 88.91% of tables in GitTablesDB, 84.25% in GitTablesSC, 93.44% in SOTAB-CTA, 76.99% in SOTAB-CPA, 25.00% in WikiTables-CTA, and 35.81% in WikiTables-CPA. This demonstrates that our retrieval and verification techniques (Sections 4 and 5) effectively identify relevant context columns for most targets. Although WikiTables-CTA and WikiTables-CPA have narrower tables (Table 2), our approach still refines context for a significant portion of tables.

Performance on Tables with Different Column Numbers. We further analyze the performance of REVEAL and REVEAL+ on tables with different column numbers, in groups of 1-4, 5-8, 9-12, 13-16, and 17+ columns, as shown in Figure 5. Observe that the baselines tend to have significant performance drop on tables with many columns, especially on wide tables with more than 16 columns, e.g., Starmie on GitTablesDB and GitTablesSC, and Watchog on

Table 4: Accuracy (%) on tables with more than 100 columns.

Dataset	Starmie	RECA	Watchog	REVEAL	REVEAL+
GitTablesDB	28.6	47.6	42.7	52.4	**66.7**
GitTablesSC	25.0	**62.5**	37.5	**62.5**	**62.5**

WikiTables-CTA and WikiTables-CPA. This is likely due to the increased complexity and noise introduced by irrelevant or redundant columns, which can hinder the model's ability to focus on the most informative features for annotation tasks. In contrast, REVEAL and REVEAL+ maintain relatively stable performance when the number of columns increases, especially when the number of columns exceeds 16 on GitTablesDB, GitTablesSC, WikiTables-CTA and WikiTables-CPA. Furthermore, in GitTablesDB and GitTablesSC, there are 21 and 8 targets in tables with 100+ columns, respectively. In Table 4, we report the performance on these extremely wide tables. Note that a small sample size may make performance sensitive to incorrect predictions. Nevertheless, REVEAL+ consistently performs the best, followed by REVEAL. The observations validate the proposed techniques in Sections 4.1 and 5, to explicitly select the most informative context columns for CTA and CPA tasks, which helps mitigate the noise introduced by irrelevant columns. This leads to more robust performance of REVEAL+ and REVEAL across different table sizes.

6.3 Efficiency Analysis

Figure 6 reports the inference time of REVEAL and REVEAL+ compared to other baselines. First, observe that REVEAL ranks top among the methods across all datasets, with the fastest inference time on SOTAB-CTA and SOTAB-CPA and similar efficiency as Doduo on the other datasets. The efficiency of REVEAL is attributed to its retrieval-based approach, which avoids the need for full-table input. More importantly, as shown in Table 3, REVEAL achieves better quality than existing methods for the annotation tasks. This indicates that REVEAL can achieve high-quality annotation with a more efficient inference process. Second, REVEAL+ incurs additional inference time due to running the verification model to further refine the column context, in order to further boost effectiveness over REVEAL. But still, the efficiency of REVEAL+ is moderate among all methods, substantially faster than RECA, Sato and TableLlama. As shown in Table 3, REVEAL+ achieves the top-1 performance on almost all settings, significantly improving the performance of REVEAL. As mentioned, in the CTA and CPA tasks, result quality is relatively more important since the inference can be done offline and time is not a critical factor. Therefore, REVEAL+ serves as a good trade-off for effectiveness over efficiency.

Top-Down Verification Inference vs. Exhaustive Verification. We evaluate the greedy top-down verification inference designed in Section 5.2 against exhaustive verification, which evaluates all possible context subsets $\mathcal{S}'$ to get $\mathcal{S}$, in terms of both accuracy and efficiency, in Figure 7. The left figure shows that the top-down strategy achieves nearly the same Macro-F1 as exhaustive search, indicating that it effectively finds the high-quality verified column context. The right figure shows that the top-down verification inference is significantly more efficient than exhaustive search. For example, on SOTAB-CPA, the top-down strategy reduces inference

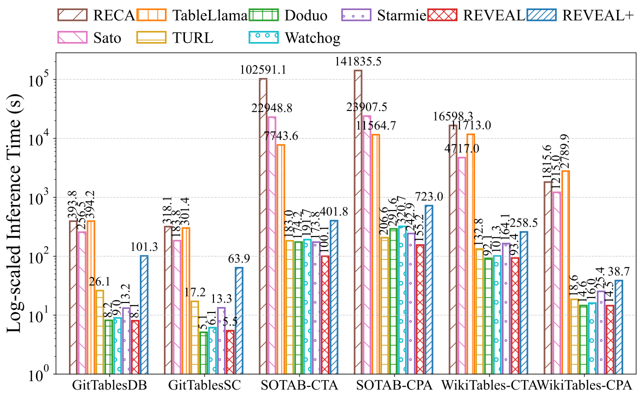


Figure 6: Inference time comparison.

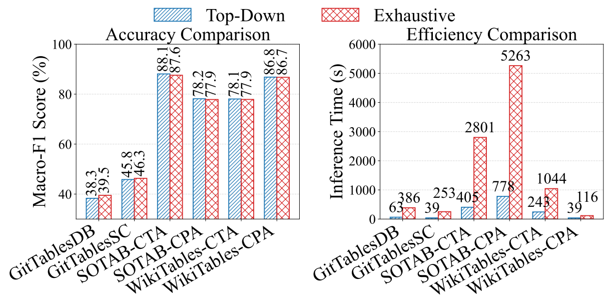


Figure 7: Comparison between the Top-Down inference strategy and Exhaustive search in terms of accuracy (Macro-F1) and inference efficiency.

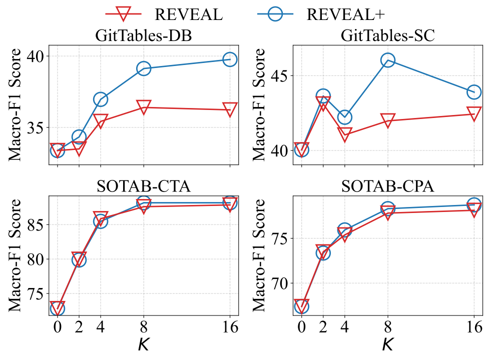


Figure 8: Vary K.

time from 5263s to 778s, achieving over 6x speedup while maintaining accuracy. Figure 7 demonstrates the effectiveness and efficiency of the proposed top-down verification inference technique.

6.4 Experimental Analysis

Varying K of Column Context Size. In the retrieval method, we retrieve a column context of K columns for a target in a table. We evaluate the impact of varying K in {0, 2, 4, 8, 16} on the performance of REVEAL and REVEAL+, as reported in Figure 8. Observe that the performance of both REVEAL and REVEAL+ improves as K increases to 8 on all datasets, except GitTablesSC with a little fluctuation. Then the performance becomes stable with further increase of K to 16. Observe that at the same K value, REVEAL+

Table 5: Ablation study in Macro-F1 (%).

	GitTablesDB	GitTablesSC	SOTAB-CTA	SOTAB-CPA
w/o encoding	30.69	39.22	87.05	77.48
TURL encoding	27.22	37.98	86.89	76.98
context-aware encoding	**36.40**	**41.98**	**87.60**	**77.81**

Table 6: Retrieval method vs. other strategies.

Micro-F1	GitTablesDB	GitTablesSC	SOTAB-CTA	SOTAB-CPA	Avg. Rank
Random	58.57	67.52	88.02	79.56	4.75
First	60.13	68.25	87.55	79.68	4.25
Nearby	60.18	68.85	<u>88.28</u>	80.78	2.75
Similar	58.47	67.86	87.87	**81.19**	4.0
Position	<u>60.40</u>	<u>69.24</u>	87.51	78.07	4.0
Ours	**61.53**	**70.9**	**88.74**	<u>80.81</u>	**1.25**

outperforms REVEAL, demonstrating the effectiveness of the verification techniques in Section 5. Figure 8 shows that using few columns with a small K may omit critical context, while increasing K beyond a moderate value yields diminishing returns. Therefore, we set $K = 8$ by default.

Ablation on Context-aware Encoding. We conduct an ablation study on the context-aware encoding in Section 4.2 by comparing REVEAL with and without the encoding, as well as with TURL encoding [6]. In Table 5, our proposed context-aware encoding achieves the best performance across all datasets. For example, on GitTablesDB, REVEAL with context-aware encoding improves Macro-F1 from 30.69% (without encoding) to 36.40% and also outperforms TURL encoding by a large margin. These results demonstrate the effectiveness of distinguishing context columns from target columns during encoding, as designed in Section 4.2.

Study on the Retrieval Technique in Section 4.1. In Section 4.1, we propose a column retrieval method that selects a compact and informative subset of K columns as the column context C. Our design emphasizes both semantic relevance and diversity, and employs MMR for context selection. We compare it with several alternative strategies. In particular, we compare the following strategies: *Random* selects K context columns at random; *First* takes the first K columns from the table; *Nearby* selects the columns adjacent to the target; *Similar* retrieves the top-K columns with the highest embedding similarity to the target column; *Position* selects the two leftmost columns and the left and right columns of the target. Note that all these methods are implemented within the same REVEAL+ framework and only differ in how to get C in Section 4.1. The results are reported in Table 6, and the improvement of ours in the last row over the others is solely achieved by Section 4.1. Observe that our method with the retrieval technique in Section 4.1 performs better than the other strategies, except *Similar* on SOTAB-CPA. This confirms the effectiveness of combining semantic relevance and diversity in column retrieval, as emphasized in our design.

Study on Equation (2). We study Equation (2) by introducing a weight λ to control the trade-off between relevance and diversity: $g(c, \tau, C) = \lambda \cos(\mathbf{e}_c, \mathbf{e}_\tau) - (1 - \lambda) \max_{c'' \in C} \cos(\mathbf{e}_c, \mathbf{e}_{c''})$. A smaller λ emphasizes diversity, while a larger λ emphasizes relevance. By default, we consider same weight for both terms, i.e., $\lambda = 0.5$. Figure 9 reports the performance of REVEAL and REVEAL+ for λ in $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. As λ increases, the performance of both methods generally improves and then plateaus or decreases. REVEAL+

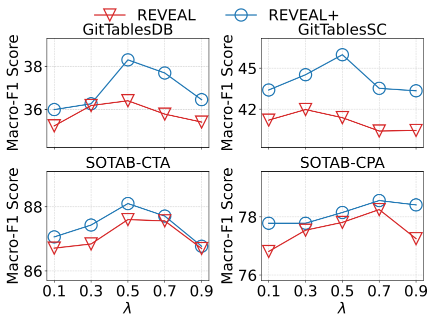


Figure 9: Vary λ.

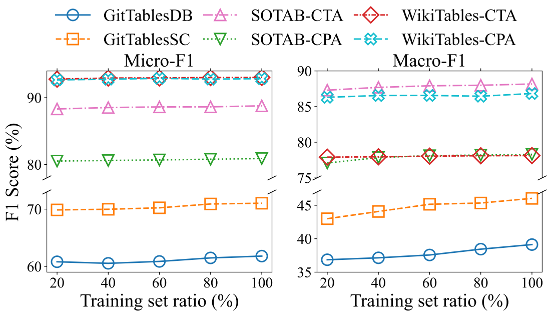


Figure 10: Learning efficiency of the verification model in REVEAL+ by varying the ratio of training data.

Table 7: REVEAL+ with different verification methods.

Macro-F1	GitTablesDB	GitTablesSC	SOTAB-CTA	SOTAB-CPA
Random	34.96	38.31	76.31	67.90
Max Confidence	34.83	34.32	76.35	69.77
Majority Voting	36.71	40.02	83.69	73.08
Weighted Voting	36.73	39.27	84.07	73.77
REVEAL+	**38.30**	**45.82**	**88.1**	**78.15**

consistently outperforms REVEAL across all datasets. Notably, REVEAL+ achieves its best performance at $\lambda = 0.5$ on most datasets, except for SOTAB-CPA, where REVEAL+ with $\lambda = 0.7$ yields even higher quality. These results indicate that it is not possible to simply tune Equation (2) to let REVEAL to achieve the same performance as REVEAL+, validating the necessity of the verification techniques in Section 5 for REVEAL+ to further enhance performance.

Learning Efficiency of the Verification Model. In Section 5, we construct a labeled dataset to train the verification model. To evaluate its learning efficiency, we vary the size of its training set from 20% to 100% and report the results of REVEAL+ in Figure 10. When the training set ratio increases, the performance of REVEAL+ generally improves in a mild way, indicating that the verification model benefits from more training data, but is not sensitive to that and can achieve good performance with limited training data. Note that on GitTablesDB and GitTablesSC, the performance of improves more significantly than on the other datasets. The reason is that GitTablesDB and GitTablesSC contain more complex tables with wider structures. Therefore, we need to have a sufficiently trained verification model to verify if a subset of columns is helpful for the target annotation tasks. Figure 10 illustrates that the verification model with the top-down inference technique in Section 5 is effective and robust to learn from training data.

Table 8: Performance of baselines with C.

	GitTablesDB		GitTablesSC		SOTAB-CTA		SOTAB-CPA	
Watchog	53.96	28.89	65.24	36.06	86.23	84.19	76.52	72.60
Watchog + C	54.27	29.32	65.35	37.06	87.90	86.47	79.66	75.91
Starmie	54.43	30.71	64.87	37.04	87.95	86.84	78.92	75.69
Starmie + C	56.98	32.11	66.88	37.47	88.29	87.49	80.40	76.78
REVEAL+	**61.53**	**38.3**	**70.9**	**45.82**	**88.74**	**88.1**	**80.81**	**78.15**

Table 9: Top-down vs. bottom-up verification inference.

	GitTablesDB		GitTablesSC		SOTAB-CTA		SOTAB-CPA	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
bottom-up	55.86	31.09	63.49	28.48	65.29	62.30	68.43	64.85
top-down	**61.53**	**38.30**	**70.90**	**45.82**	**88.74**	**88.10**	**80.81**	**78.15**

Table 10: REVEAL+ using C vs. full column set.

	GitTablesDB		GitTablesSC		SOTAB-CTA		SOTAB-CPA	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
full set	60.27	36.75	69.58	42.45	88.37	87.61	79.60	77.11
C	**61.80**	**39.12**	**71.03**	**46.04**	**88.78**	**88.17**	**80.90**	**78.30**

Study on the Verification Model. In REVEAL+, we employ a learned verification model with a top-down inference strategy to identify $\mathcal{S}$ from C. We compare it against several alternative methods: *Random*, which selects $\mathcal{S}$ from C randomly, with results averaged over 10 runs; *Max Confidence*, which selects the subset with the highest softmax probability from the prediction module f; *Majority Voting*, which determines the final label based on the most frequent label predicted across all subsets of C; and *Weighted Voting*, which averages the softmax probabilities across all subsets and selects the label with the highest mean probability. Table 7 reports the Macro-F1 results. Observe that REVEAL+ with the proposed verification model consistently outperforms all other methods across all datasets, with a significant margin. This demonstrates the effectiveness of our verification model in selecting informative column contexts for the target column, leading to improved performance in the annotation tasks.

Applying Baselines over C. We apply strong baselines, Watchog and Starmie, on the column context C retrieved in Section 4.1. As shown in Table 8, both baselines exhibit improved performance when using C, compared to using original tables. This demonstrates the effectiveness and generalizability of our idea of retrieving a compact and informative column context for the target. Nonetheless, REVEAL+ still significantly outperforms both baselines, when they use the same C, validating the effectiveness and necessity of our techniques in Section 4.2 and Section 5.

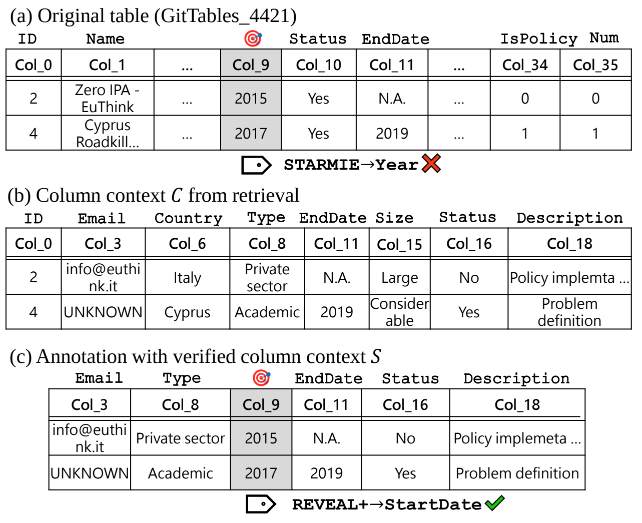
Top-down vs. Bottom-up Verification. We compare the top-down verificaiton inference in Section 5.2 with a bottom-up approach. The bottom-up approach starts with the top-ranked column in Algorithm 1 and iteratively adds columns until either all columns in C are included or the quality score from the verification model no longer improves, i.e., early stopping. Table 9 shows that bottom-up search is outperformed by our top-down approach. This supports our intuition in Section 5.2 that top-down verification, which starts from a larger high-quality context and prunes only a few noisy columns, can get better $\mathcal{S}$. In contrast, bottom-up verification may be more sensitive to the initial column choices.

Table 11: Performance on different types of columns.

	GitTablesDB			GitTablesSC		
	text	numeric	date-time	text	numeric	date-time
Starmie	49.85	70.51	53.01	56.62	80.54	58.00
REVEAL+	**56.88**	**70.78**	**59.04**	**61.76**	**84.90**	**62.00**

Table 12: Results with 512 tokens.

	GittablesDB		GittablesSC		SOTAB-CTA		SOTAB-CPA	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
Watchog	54.27	29.37	64.44	34.75	86.56	84.57	78.06	71.90
Starmie	54.16	31.44	64.85	38.31	87.92	86.91	78.77	75.77
REVEAL	<u>59.88</u>	<u>36.69</u>	<u>70.33</u>	<u>42.41</u>	<u>89.40</u>	<u>88.66</u>	<u>80.68</u>	<u>78.05</u>
REVEAL+	**62.07**	**39.56**	**71.36**	**46.35**	**89.79**	**89.18**	**80.99**	**78.16**


(a) Original table (GitTables_4421)

ID	Name		🎯	Status	EndDate		IsPolicy	Num
Col_0	Col_1	...	Col_9	Col_10	Col_11	...	Col_34	Col_35
2	Zero IPA - EuThink	...	2015	Yes	N.A.	...	0	0
4	Cyprus Roadkill...	...	2017	Yes	2019	...	1	1

STARMIE→Year ✗

(b) Column context C from retrieval

ID	Email	Country	Type	EndDate	Size	Status	Description
Col_0	Col_3	Col_6	Col_8	Col_11	Col_15	Col_16	Col_18
2	info@euthink.it	Italy	Private sector	N.A.	Large	No	Policy implemta ...
4	UNKNOWN	Cyprus	Academic	2019	Considerable	Yes	Problem definition

(c) Annotation with verified column context S

Email	Type	🎯	EndDate	Status	Description
Col_3	Col_8	Col_9	Col_11	Col_16	Col_18
info@euthink.it	Private sector	2015	N.A.	No	Policy implemeta ...
UNKNOWN	Academic	2017	2019	Yes	Problem definition

REVEAL+→StartDate ✓

Figure 11: Illustrative Example

Performance of REVEAL+ on Full Column Set. We compare REVEAL+ when verifying over the retrieved context C versus verifying over the full set of columns in a table. As shown in Table 10, REVEAL+ achieves consistently better results using C than using the full column set. This validates that the retrieval stage in REVEAL effectively eliminates irrelevant columns, enabling more focused and informative context for effective verification in REVEAL+.

Performance on Different Column Types. We analyze performance by column type: text (e.g., name, description, title), numeric (e.g., value, age, price), and datetime (e.g., date, year, time). Table 11 reports the results of REVEAL+ and the baseline Starmie for each column type. Both methods achieve higher accuracy on numeric columns, likely because numeric types are more distinct and easier to classify than text or datetime columns. Importantly, REVEAL+ consistently outperforms Starmie across all types.

Performance with 512 Tokens. We set the maximum input length of BERT to 256 tokens, following [14, 32]. In [40], it shows that varying between 256 and 512 tokens does not affect performance much. In Table 12, we report the results with 512 tokens, comparing with strong baselines. Observe that using 512 tokens yields minor improvements over 256 tokens, and our REVEAL+ and REVEAL consistently outperform the baselines. These results further demonstrate the effectiveness and robustness of our methods.

6.5 Illustrative Example

Figure 11 presents an example on a table GitTables_4421 with 136 columns from GitTablesDB, describing citizen science projects. Figure 11(a) shows a sample view of the table, where the semantic types of columns are provided for reference but are not used in the CTA task. The target column Col_9 in gray contains year-like values, such as `2015` and `2017`. Its ground truth type is `StartDate`, indicating the start date of a project, paired with the `EndDate` column Col_11. The strong baseline, Starmie, predicts the target column as `Year` when using the entire table as input. While reasonable, this prediction is incorrect. The table contains many irrelevant columns, such as Col_34 and Col_35, which do not contribute to predicting the target column type. These irrelevant columns overwhelm the baseline model, leading to suboptimal predictions. Figure 11(b) shows the column context C with 8 columns retrieved using our retrieval method in Section 4.1. These columns, such as `EndDate`, `Status`, and `Description`, are closely related to the project information and help clarify the target column's meaning. This demonstrates the retrieval method's ability to filter out irrelevant columns effectively. Figure 11(c) shows the verified column context $\mathcal{S}$ selected from C by the verification model in REVEAL+ (Section 5). Compared to C, the verification model further refines the context by removing less relevant columns, such as `Size`, ensuring only the most informative columns remain. With the verified column context, REVEAL+ correctly predicts the target column type as `StartDate`, matching the ground truth and together with the `EndDate` column, indicating a complete project timeline.

7 Related Work

Early methods for column annotation rely on hand-crafted and statistical data features. SemanticTyper [35] utilized TF-IDF for textual data and Kolmogorov-Smirnov tests [29] for numeric data to distinguish data types. Pham et al. [34] extended this approach by incorporating additional features, such as the Mann-Whitney test for numerical data and Jaccard similarity for textual data, to train logistic regression and random forest models for annotation.

Building on advancements in machine learning, recent studies have incorporated semantic features, framing CTA and CPA as multi-class classification problems. Sherlock [19] extracts multi-granularity tabular features, such as character-level, word-level, segment-level, and global-level, and trains deep learning classifiers for semantic type prediction. Sato [47] extends Sherlock by modeling table-level topics and correlations between neighbor columns.

Subsequently, language models like BERT [8] have been employed to learn representations of tabular data for table understanding [21, 45]. Column annotation methods based on language models [6, 32, 39] have gained significant attraction. For instance, TURL [6] introduces a pre-training and fine-tuning framework that uses a visibility matrix to mask irrelevant table components and generate column embeddings for downstream tasks. Doduo [39] developes a multi-task learning framework based on BERT, which takes the entire table as input and predicts column types and relations using a single model. It achieves high annotation quality by modeling token-level interactions across columns through self-attention. RECA [40] aligns schema-similar and topic-related tables with a novel named entity schema to address the complexities of wide tables and inter-table contexts. Watchog [32] employs contrastive learning techniques to tackle challenges associated with data sparsity and class imbalance in column annotation tasks, reducing reliance on high-quality annotated instances. Starmie [14], originally designed for dataset discovery in data lakes, proposes a self-supervised contrastive learning framework to train a high-quality column encoder. We adapt it to column annotation and compare its performance with our methods. These studies underscore the importance of effective representational learning for accurate annotations. However, existing models typically process all columns in a table as input for a target, relying on transformers and attention mechanisms to infer useful semantics and column interactions. This approach often fails to filter out irrelevant columns, introducing noise that can degrade performance. In contrast, our method explicitly identifies and validates contextually relevant columns for the target column, enabling a more precise understanding of column semantics. Shraga and Miller [38] propose a different problem, semantic data versioning, focusing on explaining changes between dataset versions by identifying transformations from an origin relation to a goal relation. Their Explain-Da-V method uses functional dependency (FD) discovery to select column subsets for efficient search. In contrast, we work on column annotation for tables with missing metadata—a distinct task that could benefit data versioning. Our retrieval and verification techniques in Sections 4 and 5 are technically different from the FD-based approach in [38].

Recent studies have explored the use of LLMs [12, 44, 49] for column annotation tasks [15, 31, 48]. For instance, TableLlama [48] fine-tunes LLMs on various table-related tasks, including column annotation. In our experiments, we compared our method with TableLlama and a general-purpose LLM, Qwen-Plus [44], and observed that they perform suboptimally on column annotation tasks. This suggests that the effectiveness of LLMs in such tasks depends heavily on whether they have been trained or fine-tuned on relevant table-specific corpora. Consequently, column annotation remains a challenging problem that cannot be effectively addressed by LLMs alone and requires dedicated designs and methods.

In addition, several other tasks are related to tabular data, including table discovery [7, 13, 20], such as table join search [9, 10, 50] and table union search [14, 23, 33], as well as schema matching [28, 41] and tabular data synthesis for dataset augmentation [3, 4]. These tasks rely heavily on effective table understanding and column representation learning. In future work, we plan to extend our techniques to support these broader table-related applications.

8 Conclusion

We propose a novel retrieve-and-verify framework comprising the REVEAL and REVEAL+, which selectively incorporate relevant column context to enhance annotation accuracy. REVEAL employs an unsupervised retrieval strategy to construct compact and informative column subsets, combined with context-aware encoding techniques that differentiate between target and context columns to learn effective embeddings. Building on this, REVEAL+ refines the retrieved column context using a lightweight verification model, formulating context verification as a supervised classification problem and introducing a top-down inference method to efficiently identify high-quality contexts. Extensive experiments on six benchmark datasets validate the effectiveness of our framework, with both REVEAL and REVEAL+ significantly surpassing existing state-of-the-art methods. These findings underscore the importance of selective context in table understanding and present a scalable, generalizable solution for real-world applications. As future work, a direction is to leverage the context quality scores learned by the verification model in REVEAL+ to guide the retrieval in REVEAL in a supervised manner, potentially improving performance. We also plan to extend our framework to other tabular data tasks, e.g., table generation, and explore integration with advanced large models.

References

[1] Nora Abdelmageed, Jiaoyan Chen, Vincenzo Cutrona, Vasilis Efthymiou, Oktie Hassanzadeh, Madelon Hulsebos, Ernesto Jiménez-Ruiz, Juan Sequeda, and Kavitha Srinivas. 2022. Results of SemTab 2022. In *SemTab@ISWC (CEUR Workshop Proceedings, Vol. 3320)*. CEUR-WS.org, 1–13.

[2] Dan Brickley, Matthew Burgess, and Natasha F. Noy. 2019. Google Dataset Search: Building a search engine for datasets in an open Web ecosystem. In *WWW*. ACM, 1365–1375.

[3] Jean-Flavien Bussotti, Enzo Veltri, Donatello Santoro, and Paolo Papotti. 2023. Generation of Training Examples for Tabular Natural Language Inference. *Proc. ACM Manag. Data* 1, 4 (2023), 243:1–243:27.

[4] Kuntai Cai, Xiaokui Xiao, and Graham Cormode. 2023. PrivLava: Synthesizing Relational Data with Foreign Keys under Differential Privacy. *Proc. ACM Manag. Data* 1, 2 (2023), 142:1–142:25.

[5] Jaime G. Carbonell and Jade Goldstein. 1998. The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries. In *SIGIR*. ACM, 335–336.

[6] Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2022. TURL: Table Understanding through Representation Learning. *SIGMOD Rec.* 51, 1 (2022), 33–40.

[7] Yuhao Deng, Chengliang Chai, Lei Cao, Qin Yuan, Siyuan Chen, Yanrui Yu, Zhaoze Sun, Junyi Wang, Jiajun Li, Ziqi Cao, Kaisen Jin, Chi Zhang, Yuqing Jiang, Yuanfang Zhang, Yuping Wang, Ye Yuan, Guoren Wang, and Nan Tang. 2024. LakeBench: A Benchmark for Discovering Joinable and Unionable Tables in Data Lakes. *Proc. VLDB Endow.* 17, 8 (2024), 1925–1938.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT (1)*. Association for Computational Linguistics, 4171–4186.

[9] Yuyang Dong, Kunihiro Takeoka, Chuan Xiao, and Masafumi Oyamada. 2021. Efficient Joinable Table Discovery in Data Lakes: A High-Dimensional Similarity-Based Approach. In *ICDE*. IEEE, 456–467.

[10] Yuyang Dong, Chuan Xiao, Takuma Nozawa, Masafumi Enomoto, and Masafumi Oyamada. 2023. DeepJoin: Joinable Table Discovery with Pre-trained Language Models. *Proc. VLDB Endow.* 16, 10 (2023), 2458–2470.

[11] Xingyu Du, Gongsheng Yuan, Sai Wu, Gang Chen, and Peng Lu. 2024. In Situ Neural Relational Schema Matcher. In *ICDE*. IEEE, 138–150.

[12] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024. The Llama 3 Herd of Models. *CoRR* abs/2407.21783 (2024).

[13] Grace Fan, Jin Wang, Yuliang Li, and Renée J. Miller. 2023. Table Discovery in Data Lakes: State-of-the-art and Future Directions. In *SIGMOD Conference Companion*. ACM, 69–75.

[14] Grace Fan, Jin Wang, Yuliang Li, Dan Zhang, and Renée J. Miller. 2023. Semantics-aware Dataset Discovery from Data Lakes with Contextualized Column-based Representation Learning. *Proc. VLDB Endow.* 16, 7 (2023), 1726–1739.

[15] Benjamin Feuer, Yurong Liu, Chinmay Hegde, and Juliana Freire. 2024. ArcheType: A Novel Framework for Open-Source Column Type Annotation using Large Language Models. *Proc. VLDB Endow.* 17, 9 (2024), 2279–2292.

[16] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On Calibration of Modern Neural Networks. In *ICML (Proceedings of Machine Learning Research, Vol. 70)*. PMLR, 1321–1330.

[17] Dan Hendrycks and Kevin Gimpel. 2017. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. In *ICLR (Poster)*. OpenReview.net.

[18] Madelon Hulsebos, Çagatay Demiralp, and Paul Groth. 2023. GitTables: A Large-Scale Corpus of Relational Tables. *Proc. ACM Manag. Data* 1, 1 (2023), 30:1–30:17.

[19] Madelon Hulsebos, Kevin Zeng Hu, Michiel A. Bakker, Emanuel Zgraggen, Arvind Satyanarayan, Tim Kraska, Çagatay Demiralp, and César A. Hidalgo. 2019. Sherlock: A Deep Learning Approach to Semantic Data Type Detection. In *KDD*. 1500–1508.

[20] Madelon Hulsebos, Wenjing Lin, Shreya Shankar, and Aditya G. Parameswaran. 2024. It Took Longer than I was Expecting: Why is Dataset Search Still so Hard?. In *HILDA@SIGMOD*. ACM, 1–4.

[21] Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. 2021. TABBIE: Pretrained Representations of Tabular Data. In *NAACL-HLT*. Association for Computational Linguistics, 3446–3456.

[22] Ernesto Jiménez-Ruiz, Oktie Hassanzadeh, Vasilis Efthymiou, Jiaoyan Chen, and Kavitha Srinivas. 2020. SemTab 2019: Resources to Benchmark Tabular Data to Knowledge Graph Matching Systems. In *ESWC (Lecture Notes in Computer Science, Vol. 12123)*. Springer, 514–530.

[23] Aamod Khatiwada, Grace Fan, Roee Shraga, Zixuan Chen, Wolfgang Gatterbauer, Renée J. Miller, and Mirek Riedewald. 2023. SANTOS: Relationship-based Semantic Table Union Search. *Proc. ACM Manag. Data* 1, 1 (2023), 9:1–9:25.

[24] Aamod Khatiwada, Roee Shraga, Wolfgang Gatterbauer, and Renée J. Miller. 2022. Integrating Data Lake Tables. *Proc. VLDB Endow.* 16, 4 (2022), 932–945.

[25] Keti Korini and Christian Bizer. 2023. Column type annotation using chatgpt. *arXiv preprint arXiv:2306.00745* (2023).

[26] Keti Korini and Christian Bizer. 2024. Column property annotation using large language models. In *European Semantic Web Conference*. Springer, 61–70.

[27] Keti Korini, Ralph Peeters, and Christian Bizer. 2022. SOTAB: The WDC Schema.org Table Annotation Benchmark. In *SemTab@ISWC (CEUR Workshop Proceedings, Vol. 3320)*. CEUR-WS.org, 14–19.

[28] Christos Koutras, George Siachamis, Andra Ionescu, Kyriakos Psarakis, Jerry Brons, Marios Fragkoulis, Christoph Lofi, Angela Bonifati, and Asterios Katsifodimos. 2021. Valentine: Evaluating Matching Techniques for Dataset Discovery. In *ICDE*. IEEE, 468–479.

[29] Erich Leo Lehmann and Joseph P. Romano. 2008. Testing Statistical Hypotheses, Third Edition. (2008).

[30] Aristotelis Leventidis, Laura Di Rocco, Wolfgang Gatterbauer, Renée J. Miller, and Mirek Riedewald. 2023. DomainNet: Homograph Detection and Understanding in Data Lake Disambiguation. *ACM Trans. Database Syst.* 48, 3 (2023), 9:1–9:40.

[31] Peng Li, Yeye He, Dror Yashar, Weiwei Cui, Song Ge, Haidong Zhang, Danielle Rifinski Fainman, Dongmei Zhang, and Surajit Chaudhuri. 2024. Table-GPT: Table Fine-tuned GPT for Diverse Table Tasks. *Proc. ACM Manag. Data* 2, 3 (2024), 176.

[32] Zhengjie Miao and Jin Wang. 2023. Watchog: A Light-weight Contrastive Learning based Framework for Column Annotation. *SIGMOD* 1, 4 (2023), 272:1–272:24.

[33] Fatemeh Nargesian, Erkang Zhu, Ken Q. Pu, and Renée J. Miller. 2018. Table Union Search on Open Data. *Proc. VLDB Endow.* 11, 7 (2018), 813–825.

[34] Minh Pham, Suresh Alse, Craig A. Knoblock, and Pedro A. Szekely. 2016. Semantic Labeling: A Domain-Independent Approach. In *ISWC (1) (Lecture Notes in Computer Science, Vol. 9981)*. 446–462.

[35] S. K. Ramnandan, Amol Mittal, Craig A. Knoblock, and Pedro A. Szekely. 2015. Assigning Semantic Labels to Data Sources. In *ESWC (Lecture Notes in Computer Science, Vol. 9088)*. Springer, 403–417.

[36] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP/IJCNLP (1)*. Association for Computational Linguistics, 3980–3990.

[37] Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. 2023. Large Language Models Can Be Easily Distracted by Irrelevant Context. In *ICML (Proceedings of Machine Learning Research, Vol. 202)*. PMLR, 31210–31227.

[38] Roee Shraga and Renée J. Miller. 2023. Explaining Dataset Changes for Semantic Data Versioning with Explain-Da-V. *Proc. VLDB Endow.* 16, 6 (2023), 1587–1600.

[39] Yoshihiko Suhara, Jinfeng Li, Yuliang Li, Dan Zhang, Çagatay Demiralp, Chen Chen, and Wang-Chiew Tan. 2022. Annotating Columns with Pre-trained Language Models. In *SIGMOD Conference*. ACM, 1493–1503.

[40] Yushi Sun, Hao Xin, and Lei Chen. 2023. RECA: Related Tables Enhanced Column Semantic Type Annotation Framework. *Proc. VLDB Endow.* 16, 6 (2023), 1319–1331.

[41] Jianhong Tu, Ju Fan, Nan Tang, Peng Wang, Guoliang Li, Xiaoyong Du, Xiaofeng Jia, and Song Gao. 2023. Unicorn: A Unified Multi-tasking Model for Supporting Matching Tasks in Data Integration. *Proc. ACM Manag. Data* 1, 1 (2023), 84:1–84:26.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *NIPS*. 5998–6008.

[43] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *EMNLP (Demos)*. Association for Computational Linguistics, 38–45.

[44] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024. Qwen2 Technical Report. *CoRR* abs/2407.10671 (2024).

[45] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. In *ACL*. Association for Computational Linguistics, 8413–8426.

[46] Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2024. Making Retrieval-Augmented Language Models Robust to Irrelevant Context. In *ICLR*. OpenReview.net.

[47] Dan Zhang, Yoshihiko Suhara, Jinfeng Li, Madelon Hulsebos, Çagatay Demiralp, and Wang-Chiew Tan. 2020. Sato: Contextual Semantic Type Detection in Tables. *Proc. VLDB Endow.* 13, 11 (2020), 1835–1848.

[48] Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 2024. TableLlama: Towards Open Large Generalist Models for Tables. In *NAACL-HLT*. Association for Computational Linguistics, 6024–6044.

[49] Justin Zhao, Timothy Wang, Wael Abid, Geoffrey Angus, Arnav Garg, Jeffery Kinnison, Alex Sherstinsky, Piero Molino, Travis Addair, and Devvret Rishi. 2024. LoRA Land: 310 Fine-tuned LLMs that Rival GPT-4, A Technical Report. *CoRR* abs/2405.00732 (2024).

[50] Erkang Zhu, Dong Deng, Fatemeh Nargesian, and Renée J. Miller. 2019. JOSIE: Overlap Set Similarity Search for Finding Joinable Tables in Data Lakes. In *SIGMOD Conference*. ACM, 847–864.