Automatic Table Union Search with Tabular Representation Learning

Xuming Hu[1*], **Shen Wang**[2], **Xiao Qin**[2], **Chuan Lei**[2], **Zhengyuan Shen**[2],
Christos Faloutsos[2], **Asterios Katsifodimos**[2], **George Karypis**[2], **Lijie Wen**[1], **Philip S. Yu**[1,3]
[1]Tsinghua University,[2]Amazon Web Services,[3]University of Illinois at Chicago
hxm19@mails.tsinghua.edu.cn,wenlj@tsinghua.edu.cn,psyu@uic.edu
{shenwa,drxqin,chuanlei,donshen}@amazon.com
{faloutso,akatsifo,gkarypis}@amazon.com

Abstract

Given a data lake of tabular data as well as a query table, how can we retrieve all the tables in the data lake that can be *unioned* with the query table? Table union search constitutes an essential task in data discovery and preparation as it enables data scientists to navigate massive open data repositories. Existing methods identify uniability based on *column representations* (word surface forms or token embeddings) and column relation represented by column representation similarity. However, the semantic similarity obtained between column representations is often insufficient to reveal latent relational features to describe the column relation between pair of columns and not robust to the table noise. To address these issues, in this paper, we propose a multi-stage self-supervised table union search framework called AUTOTUS, which represents column relation as a vector– *column relational representation* and learn column relational representation in a multi-stage manner that can better describe column relation for table unionability prediction. In particular, the large language model powered contextualized column relation encoder is updated by adaptive clustering and pseudo label classification iteratively so that the better *column relational representation* can be learned. Moreover, to improve the robustness of the model against table noises, we propose table noise generator to add table noise to the training table data. Experiments on real-world datasets and synthetic test set augmented with table noise show that AUTOTUS achieves 5.2% performance gain over the SOTA baseline.

1 Introduction

The growing availability of tabular data from academia and industry brings new opportunities for economic growth and social benefit, which has also attracted considerable attentions in the natural language processing (NLP) community. A number of work has been done, such as Table QA (Chen et al., 2020; Zhu et al., 2021), Table fact verification (Aly et al., 2021; Guo et al., 2022), Table summarization (Xi et al., 2023), Table understanding (Yang et al., 2022; Liu et al., 2022a, 2023), etc. More recently, table union search (Nargesian et al., 2018; Bogatu et al., 2020; Khatiwada and Fan, 2023) is proposed to facilitate tabular dataset discovery applications (Brickley et al., 2019; Galhotra and Khurana, 2020; Santos et al., 2022), which can benefit the tabular data integration and analysis. The table union search aims to find all tables in a data lake that have the columns from the same domain as the query table. With the help of the table union search, the understanding of the tabular data is largely improved and it can potentially benefit other tabular-focused NLP downstream tasks. Therefore, table union search is a non-trival NLP research problem.

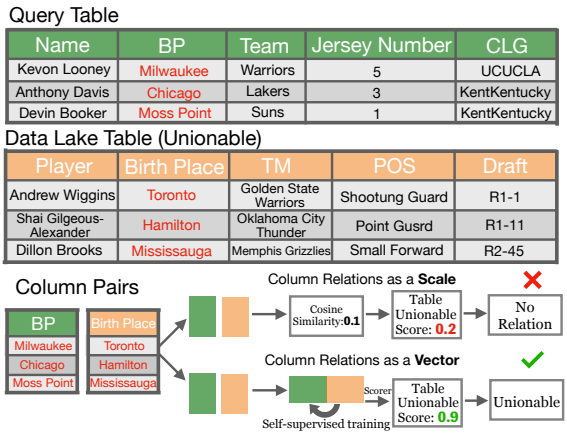


Figure 1: The table union search example on Open Data.

Recent literature shows that column representation methods are useful for table union search. The basic idea is to map the column to a latent vector space (e.g. column representations (Bogatu et al., 2020; Chepurko et al., 2020) or token embeddings (Zhang and Balog, 2017; Herzig et al., 2020; Fan et al., 2022; Yang et al., 2022)) and then compute the table unionability score based on the column relation obtained from the similarity scores of col-

*Work done during an internship at Amazon Web Services.

Findings of the Association for Computational Linguistics: ACL 2023, pages 3786–3800
July 9-14, 2023

umn pairs between query table and target tables. However, these methods only consider the *column representation* and their similarity as the column relation to compute the table unionability score, which is not enough to describe the relationship between two columns and may cause incorrect prediction. Figure 1 shows an example of the table union search with a query table and a unionable table. By looking at the second columns (in red) of two tables, the computed column similarity score (a scalar) from column embedding is low due to the dissimilar token embeddings. In this case, the column relation is not intuitive, taking only the similarity between column embeddings to describe the column relation is not enough to predict the table unionability accurately. However, if we combine each column representation and model them as an embedding (a vector), the new representation (column relational representation) is more informative and can better describe the unintuitive column relation to make the table unionability prediction more accurately. Therefore, a better approach is needed, which can make the table unionability prediction via the column relational representation.

On the other hand, real-world table noises widely exist (Koutras et al., 2021; Liu et al., 2022b). Taking column "CLG" in the query table and column "POS" in the data lake table as examples, both columns have abbreviated column names, tokens are repeated in column values (e.g., "UCLA" becomes "UCUCLA") and proximal characters in keyboard are replaced (e.g., "Shooting Guard" becomes 'Shootung Guard"). Computing column representations or column relational representations based on token embeddings with tabular noise introduces unavoidable wrong predictions. Therefore, it requires the table union search model to be robust to the table noise during optimization.

To address above issues, we propose a multi-stage self-supervised table union search framework called AUTOTUS, which represents column relation as a vector– *column relational representation* and learn column relational representation in a multi-stage manner that can better describe column relation for unionability prediction. In particular, the large language model powered contextualized column relation encoder is updated by adaptive clustering and pseudo label classification iteratively so that the better *column relational representation* can be learned. Moreover, to improve the robustness of the model against table noises, we propose table noise generator to add table noise to the training table data. Extensive experiments are conducted on three real-world datasets and show significant performance gains compared to existing state-of-the-art methods and demonstrates the robustness of AUTOTUS against the table noise.

Our contributions are four-folded: **1)** We propose a novel framework AUTOTUS that can leverage column relational representation for table union search instead of pairwise column similarity. **2)** We propose a multi-stage self-supervised method that can update the large language model powered contextualized column relation encoder in a step by step and iterative way to learn better column relational representation. **3)** We propose a table noise generator and add table noise to the training tabular data to improve the robustness of the model against the table noise. **4)** We conduct extensive experiment on real-world datasets as well as synthetic test set augmented with table noise and demonstrate significant performance gain of AUTOTUS over the strong baselines and shows the robustness of AUTOTUS against the table noise.

2 Problem Definition

Table union search discovers all tables in a set of data lake tables that are unionable with the query table. Specifically, we focus on the automatically case where no labeled training data is available. We define a set of data lake tables as $\mathcal{T}$. For each table $T \in \mathcal{T}$ which consists of m columns $\{t_1, t_2, \cdots, t_m\}$. There is a column relation $e_{ij} \in \mathcal{E}$ for each pairs of columns t_i, t_j that describes their relationship. We employ a contextualized column relational encoder to obtain the column relational representation $\mathbf{z}_{e_{ij}} \in \mathcal{Z}$ for each column relation e_{ij}. The table unionability score $u_{(T_i,T_j)} \in \mathcal{U}$ is obtained by the table scorer that goes over all column relational representations between two tables T_i, T_j. Similar to previous studies (Nargesian et al., 2018; Bogatu et al., 2020; Khatiwada and Fan, 2023), given a query table Q, table union search retrieves the top-y unionable tables $\mathcal{Q} \subseteq \mathcal{T}$, where $|\mathcal{Q}| = y$ and $\forall T \in \mathcal{Q}$ and $T^{'} \in \mathcal{T} - \mathcal{Q}$, we have $u_{(Q,T)} \geq u_{(Q,T')}$.

3 Proposed Method

In this section, we introduce the proposed AUTOTUS. The framework (as shown in Figure 2) consists of following components: Multi-stage Self-supervised Column Relational Representation

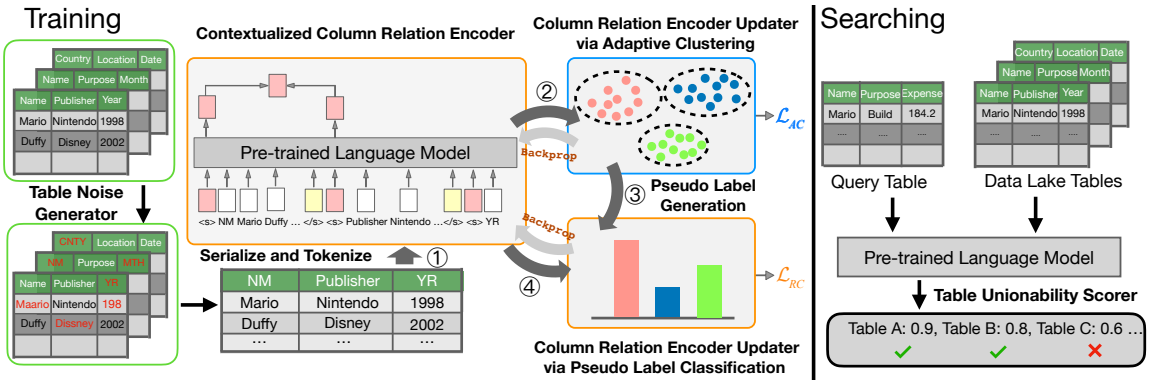


Figure 2: Overview of AUTOTUS framework.

Learning, Table Noise Generation and Table Unionability Scorer. We first illustrate the motivation for AUTOTUS: The existing column representation methods use cosine similarity to measure the unionability of column pairs. The column encoder in these methods is usually only trained with a **binary** classification task, i.e. whether two columns are unionable. In AUTOTUS, the table encoder can benefit from an additional task: the identification of hidden semantic types (domains, task knowledge, etc.) from the column pairs. In the absence of explicit type labels, we use adaptive clustering on column pairs and learn those hidden types. The classification step is then used to update the table encoder, so that the embeddings are generated in a way that minimizes the hidden type prediction errors. In short, the clustering and classification modules learn the hidden unionable semantic types, resulting in a better table encoder for the table union search task.

3.1 Multi-stage Self-supervised Column Relational Representation Learning

The Multi-stage Self-supervised Column Relational Representation Learning module aims to extract column relation as column relational representation and learn column relational representation in multi-stage manner that can better describe column relation for unionability prediction. As illustrated in Figure 2, this module consists of three components: Contextualized Column Relation Encoder, Column Relation Encoder Updater via Adaptive Clustering, and Column Relation Encoder Updater via Pseudo Label Classification. The Contextualized Column Relation Encoder serializes and tokenizes the column relations as the input and leverages the pretrained BERT (Devlin et al., 2019) to generate contextualized column relation embeddings. The Column Relation Encoder Updater via Adaptive Clustering optimizes Contextualized Column Relation Encoder through relaxed constraint to obtain better column relational representation and generate the cluster label as the pseudo label. The Column Relation Encoder Updater via Pseudo Label Classification uses the generated pseudo label to further update the Contextualized Column Relation Encoder via more strict constraint to generate improved column relational representation. The above two updates are performed in a bootstrapped loop that iteratively improve the language model to generate better column representations.

Contextualized Column Relation Encoder Contextualized Column Relation Encoder extracts the contextualized column relation embeddings to represent the relational features between two columns. We leverage a large language model, BERT (Devlin et al., 2019), to effectively encode column relations, along with their context tabular information.

The pre-trained large language models (such as BERT) support an input length of up to 512 tokens (Devlin et al., 2019). However, columns in a real-world table may contain thousands of tokens, and randomly discarding tokens will lead to loss of important semantics. Therefore, in this work, we score the importance of each token or cell in a column, and keep cells based on their importance. More specifically, we adopt TF-IDF score to calculate the token importance through its inverse document frequency: $\log(|\mathcal{Y}|)/\{t|\text{ token} \in t, t \in \mathcal{Y}\}$, where t is a column and $|\mathcal{Y}|$ is the number of all data lake columns. Then the cell score is calculated by averaging the TF-IDF of all tokens within the cell. After obtaining the importance score of each column-level cell, the row alignment can ensure the correctness of the table semantics, e.g.: Jimmy Butler's birthplace is aligned to the United States instead of the United Kingdom. Therefore, we cal-

culate the average score of cells in each row and sort them in descending order. Next, we select rows until the total token count reaches the input budget and obtain the table: $T = [t_1, t_2, \cdots, t_m]$.

As shown in Figure 2, we serialize the table by column and follow the marking schema adopted in Soares et al. (2019) to augment T with two reserved tokens to mark the start and the end of each column. We introduce $\langle s \rangle$ and $\langle /s \rangle$ and inject them into T:

$$T = \big[\langle s \rangle, t_1, \langle /s \rangle, \cdots, \langle s \rangle, t_m, \langle /s \rangle\big], \quad (1)$$

as the input token sequence for the encoder.

We denote the Contextualized Column Relation Encoder as $f_\theta(T, \langle s \rangle, \langle /s \rangle)$. Instead of using the output of $[CLS]$ token from BERT itself which summarizes the whole table-level semantics, we get the contextualize column relation semantics by concatenating the representations $\langle s \rangle$ of any two columns in different tables or in the same table, and derive a fixed-length representation $\mathbf{z}_{e_{ij}} \in \mathbb{R}^{2 \cdot z_R}$:

$$\mathbf{z}_{e_{ij}} = [\mathbf{c}_{\langle s_{t_i} \rangle}, \mathbf{c}_{\langle s_{t_j} \rangle}]. \quad (2)$$

Column Relation Encoder Updater via Adaptive Clustering The Column Relation Encoder Updater via Adaptive Clustering encourages each low-confidence relation label distribution to be closer to the high-confidence cluster centroids, improves the confidence of the clusters, and optimizes Contextualized Column Relation Encoder by relax-constraints to generate better column relational representations. In addition, the cluster label generated for each column relation could be treated as self-supervision signal, which serves as the pseudo label for language model update.

More specifically, after obtaining N column relational representations $\mathcal{Z} = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_N\}$, Column Relation Encoder Updater via Adaptive Clustering computes K cluster centroids $\{\xi_k \in \mathbb{R}^{h_{AC}}\}_{k=1}^{K}$, and softly assigns all N column relation representations to K clusters. In practice, we first adopt standard k-means clustering among N column relation data points $\mathbb{R}^{h_{AC}}$ to obtain K initial cluster centroids $\{\xi_k \in \mathbb{R}^{h_{AC}}\}_{k=1}^{K}$. Inspired by Maaten and Hinton (2008); Hu et al. (2020), we leverage the Student's t-distribution as the kernel to measure the similarity of the embedding point $\mathbf{z}_n$ to each centroid ξ_k as:

$$q_{nk} = \frac{(1 + ||\mathbf{z}_n - \xi_k||^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{k'=1}^{K}(1 + ||\mathbf{z}_n - \xi_{k'}||^2/\alpha)^{-\frac{\alpha+1}{2}}}, \quad (3)$$

where q_{nk} can be viewed as the probability of assigning column relation embedding $\mathbf{z}_n$ to the cluster k as the soft assignment and α denotes the degree of the Student's t-distribution. We set $\alpha = 1$ for all experiments.

We normalize each cluster by adopting the frequency as an auxiliary target distribution proposed by Xie et al. (2016) in Equation 4 and iteratively refine each cluster with the help of the auxiliary target distribution:

$$p_{nk} = \frac{q_{nk}^2/f_k}{\sum_{k'} q_{nk'}^2/f_{k'}}, \quad (4)$$

where $f_k = \sum_{n=1}^{N} q_{nk}, k = 1, 2, \cdots, K$ is the soft cluster frequency. With the auxiliary target distribution, we could encourage each cluster to learn from its high confidence cluster assignments and simultaneously alleviating the bias caused by imbalanced clusters. We define KL divergence loss between the soft assignments q_n and the auxiliary distribution p_n to train the Contextualized Column Relation Encoder as follows:

$$\mathcal{L}_{AC} = KL(P||Q) = \sum_n \sum_k p_{nk} log \frac{p_{nk}}{q_{nk}}. \quad (5)$$

We adopt the label associated with the largest probability as the pseudo label μ_n for the n-th column relation:

$$\mu_n = \underset{k \in K}{\operatorname{argmax}}\, p_{nk}. \quad (6)$$

In summary, comparing with traditional clustering methods such as k-means, Column Relation Encoder Updater via Adaptive Clustering adopts an iterative, soft-assignment learning process which encourages high-confidence assignments and uses high-confidence assignments to improve low confidence ones. Therefore, Column Relation Encoder Updater via Adaptive Clustering is robust enough to different cluster numbers K, thus solving the situation that we have no prior knowledge of specific domains and the relational distributions in the column relational representations. To mitigate the possible negative impact of choosing a unideal initial centroid, Column Relation Encoder Updater via Adaptive Clustering randomly re-selects a set of K initial centroids if $\mathcal{L}_{AC}$ does not decrease after the first epoch.

Column Relation Encoder Updater via Pseudo Label Classification Although the clustering module forces the relational label distribution of

low confidence to be close to the cluster centroids of high confidence to improve the confidence of clustering. However, the clusters may contain some different labels, which affects the clustering purity. Therefore, we utilize the Column Relation Encoder Updater via Pseudo Label Classification module to classify the pseudo labels, and obtain better column relational representations by improving the clustering purity. In practice, the Column Relation Encoder Updater via Adaptive Clustering generates cluster labels $M = \{\mu_1, \mu_2, ..., \mu_N\}$ for all column relations as pseudo labels. Column Relation Encoder Updater via Pseudo Label Classification could adopt these pseudo labels as self-supervisions derived from the corpora themselves and use pseudo labels to guide the feature representation learning in Contextualized Column Relation Encoder as well as column relation classifier learning in Column Relation Encoder Updater via Pseudo Label Classification.

The Column Relation Encoder Updater via Pseudo Label Classification module learns to predict the pseudo labels as golden labels. More specifically, we have:

$$\mathbf{l}_n = \tau_\phi(f_\theta(T, \langle s\rangle, \langle /s\rangle)), \tag{7}$$

where τ_ϕ denotes the column relation classification module with parameters ϕ and $\mathbf{l}_n$ is the probability distribution for the n-th sample over K pseudo labels. To find the best performance parameters θ for Contextualized Column Relation Encoder and ϕ for the classifier, we optimize the classification loss:

$$\mathcal{L}_{RC} = \min_{\theta,\phi} \frac{1}{N} \sum_{n=1}^{N} loss(\mathbf{l}_n, \text{one\_hot}(\mu_n)), \tag{8}$$

where $loss$ is the cross entropy loss and $\text{one\_hot}(\mu_n)$ returns a one-hot pseudo label assignment vector.

3.2 Table Noise Generator

The table noise generator module aims to improve the robustness of the model against the table noise. We firstly summarizes the noise types that exist in the real-world tabular data, and introduce these table noises to the column relations during the self-supervised training. Specifically, we defined two types of table noise: **column value noise** an **column name noise**. Here are some representative column value noises: 1) Replace / insert characters with proximal characters, e.g., computer → cimputer / coimputer. 2) Delete / repeat characters, e.g., computer → compuer / compputer. 3) Change the numeral display format like scientific notation, e.g., 12000 → 1.2e4. Here are some representative column name noises: 1) Prefix column names with table name, e.g., travel_destination → tourism_statistics_travel_destination. 2) Abbreviate column names, e.g., travel_destination → tra_dest. 3) Drop vowels, e.g., travel_destination → trvl_dstntn. 4) Synonym substitution (if exists), e.g., travel_destination → tourist_destination. 5) Acronym, e.g., travel_destination → t_d.

For each table T in the data lake tables $\mathcal{T}$, we randomly apply above noise to the training tabular data to improve the robustness of our model against the table noise.

3.3 Table Unionability Scorer

The Table Scorer obtains the table unionability score based on the column relational representations between column relations in the query table and data lake tables, which can provide more sufficient information to clarify whether the two tables come from the same domain and have a union relation. After we obtained the Contextualized Column Relation Encoder with discriminative power to get the column relation embeddings $\mathbf{z}_{e_{ij}} = (\mathbf{c}_{\langle s_{t_i}\rangle}, \mathbf{c}_{\langle s_{t_j}\rangle})$, we adopt the Column Relation Encoder Updater via Pseudo Label Classification to assign the K relational labels to all N column relations $\mathbf{z}^n = (\mathbf{z}_{e_{ij}})^n$ in the two tables:

$$\psi_{nk} = \underset{k\in K, n\in N}{\text{argmax}}\ \tau_\phi(\mathbf{z}^{nk}), \tag{9}$$

and assign the majority label κ as the predict relational label for the query table and data lake table. We count the number of majority labels κ in the all N column relational representations between two tables as M, so the table unionability score is M/N. This score no longer depends on the calculation of cosine similarity between two column features to obtain the column unionability score, but obtains table unionability score through the column unionability scores. The higher the score, the greater the probability of these two tables being unionable. Instead of computational methods with insufficient column relational information, AUTOTUS directly model the complex column relational representations, which can better address the table union search task.

Test / Train	# Tables	# Columns	Avg. # Rows	Size (GB)
SANTOS Small	550	6,322	6,921	0.45
TUS Small	1,530	14,810	4,466	1
TUS Large	5,043	54,923	1,915	1.5
SANTOS Small	550	6,162	7,732	0.5
TUS Small	1,530	13,926	4,032	0.9
TUS Large	5,043	55,462	1,729	1.3

Table 1: The statistics of datasets for three benchmarks.

4 Experimental Evaluation

We conduct extensive experiments on real-world public datasets as well as a synthetic test set augmented with table noise to show the effectiveness of our AUTOTUS on table union search task, and give a detailed analysis.

4.1 Experimental Setups

Datasets. To evaluate our technique, three public datasets are used : **SANTOS** (Khatiwada and Fan, 2023), **TUS small**, and **TUS large** (Nargesian et al., 2018). These three datasets are subsets of Open Data released by authorities such as US Open Data[1], UK Open Data[2], and Canada Open Data[3] with manually labeled data. However, these data are usually officially filtered to remove a lot of table noise to maintain high quality. In order to verify the robustness of the model, we synthesized a test set with table noise, we added both column value and column name noises to the tabular data and obtained the synthesized **SANTOS**$_N$ and **TUS large**$_N$. Note that although we add table noises, the human-annotated labels remain unchanged. To avoid information leakage, the manually labeled dataset is not used during the training. We randomly generated the same number of unlabeled training tables as the labeled tables from Open Data (Koutras et al., 2021; Khatiwada and Fan, 2023). We give the detailed statistics of datasets for three benchmarks in Table 1. We also give the dataset construction details in Appendix A.

Experimental Settings and Metrics. We adopt `BERT-Base`, `BERT-Large` (Devlin et al., 2019) and `RoBERTa-Large` (Liu et al., 2019) as the encoder and set max-length as 512. For Column Relation Encoder Updater via Adaptive Clustering, we adopt k-means and set $K = 50$ to get the initial centroids. We stop the clustering and classification loop when the current pseudo labels differs by less than 10% from the previous epoch. For Column Relation Encoder Updater via Pseudo Label Classification, we adopt the fully connected layer as τ_ϕ and set dropout rate as 10%, learning rate as $1e{-}5$ and warmup as 0.1. To allow fully connected layer to warm up, we fixed the parameters in f_θ for the first two epoches. We add one noise of column data or column schemata to each column to obtain the training dataset. We first randomly choose whether to add noise in the column data or column schemata, and secondly, for all replacement, addition, or deletion of noise at the token level, we select 20% of all tokens in the cell of the column for modification.

For evaluation metrics, following the previous work (Bogatu et al., 2020; Khatiwada and Fan, 2023), we adopt the Mean Average Precision at Y ($MAP@Y$) and Recall at Y ($R@Y$) to evaluate the effectiveness of the searched top-Y table results. Note that $MAP@Y$ is the average value of Precision at y ($P@y$), where $y = 1, 2, \cdots, Y$. Formally, given the query table Q and a set of data lake tables $\mathcal{T}$, we define $\mathcal{T}_Q$ as the set of unionable tables based on the ground truth and $\mathcal{T}_Q'$ as the set of top-Y unionable table results using the searching methods. The $P@Y$ and $R@Y$ are calculated as:

$$P@Y = \frac{\mathcal{T}_Q \cap \mathcal{T}_Q'}{\mathcal{T}_Q'}, R@Y = \frac{\mathcal{T}_Q \cap \mathcal{T}_Q'}{\mathcal{T}_Q} \quad (10)$$

Note that perfect $R@Y$ is not possible when the ground truth contains less than Y searched table results. We define the Mean Average Precision $MAP@Y$ as:

$$MAP@Y = \frac{1}{Y}\sum_{y=1}^{Y} P@y \quad (11)$$

For a fair comparison, we adopt the $Y = 10$ on the SANTOS benchmarks and $Y = 60$ on the TUS benchmarks to be consistent with Nargesian et al. (2018); Khatiwada and Fan (2023).

Baseline Models. We compare AUTOTUS with baseline models in two categories. The first category is **table union search model** that adopts various column features to calculate the table unionability score. We adopt **D^{3}L** (Bogatu et al., 2020), **SATO** (Zhang et al., 2020), **Starmie** (Fan et al., 2022), and **SANTOS** (Khatiwada and Fan, 2023) as baselines. The second category is **table pre-trained model** that leverages unlabeled tables for

[1] https://data.gov/
[2] https://www.data.gov.uk/
[3] https://open.canada.ca/en/open-data

Methods	SANTOS		TUS small		TUS large		SANTOS$_N$		TUS large$_N$	
	MAP@10	R@10	MAP@60	R@60	MAP@60	R@60	MAP@10	R@10	MAP@60	R@60
BERT-Base (Devlin et al., 2019)	79.5	53.7	74.1	16.3	63.1	13.4	69.5	46.9	55.3	9.4
TaBERT (Yin et al., 2020)	83.6	56.2	80.3	18.7	76.4	15.8	74.8	50.7	67.3	11.4
TABBIE (Iida et al., 2021)	86.3	58.5	82.4	19.2	78.7	15.3	76.9	51.6	65.2	11.8
TUTA (Wang et al., 2021)	86.7	59.2	84.5	20.1	80.5	16.9	77.9	52.3	71.6	13.2
FORTAP (Cheng et al., 2022)	88.5	60.9	87.3	20.6	81.3	17.0	78.7	52.7	72.8	12.8
TableFormer (Yang et al., 2022)	89.4	61.4	86.9	20.2	83.2	17.3	79.2	53.5	72.7	12.9
D^{3}L (Bogatu et al., 2020)	82.8	56.3	79.5	19.7	66.3	15.2	73.5	51.0	57.8	9.7
SATO (Zhang et al., 2020)	92.7	68.2	91.7	21.5	86.7	18.4	80.1	54.5	77.3	14.5
Starmie† (Fan et al., 2022)	93.9	69.0	94.8	23.3	89.5	18.9	81.0	55.1	78.4	14.3
SANTOS(Khatiwada and Fan, 2023)	94.3	69.2	85.7	21.0	–	–	81.4	55.3	–	–
AutoTUS$_{\texttt{BERT-Base}}$	**98.5**±0.4	**72.4**±0.2	**97.7**±0.5	**25.2**±0.3	**94.6**±0.7	**21.6**±0.2	**87.6**±0.5	**59.7**±0.3	**84.9**±0.6	**17.3**±0.3
w/o Table Noise Generator	96.6±0.5	71.2±0.4	96.1±0.6	24.5±0.3	92.5±0.7	20.8±0.4	83.0±0.7	56.4±0.4	80.7±0.8	14.9±0.3
w/o Encoder Update via Adaptive Clustering	94.5±0.6	71.9±0.4	94.3±0.7	23.3±0.5	91.0±0.5	20.4±0.3	83.2±0.6	56.3±0.4	80.5±0.7	14.7±0.3
w/o Encoder Update via Pseudo Label Classification	81.8±0.6	54.7±0.4	80.9±0.9	16.8±0.4	77.3±0.7	17.0±0.4	72.1±0.8	50.2±0.6	68.5±0.7	12.2±0.2
AutoTUS$_{\texttt{BERT-Large}}$	99.1±0.5	73.8±0.2	98.3±0.5	26.0±0.2	95.8±0.6	22.5±0.3	89.4±0.6	61.0±0.3	86.1±0.7	17.9±0.2
AutoTUS$_{\texttt{RoBERTa-Large}}$	99.4±0.4	73.9±0.2	98.9±0.3	26.7±0.2	96.2±0.5	22.3±0.2	90.0±0.5	60.5±0.3	86.9±0.6	18.1±0.3

Table 2: MAP@Y and R@Y comparisons (%). Results of AutoTUS are averaged over five runs. †means we replace the base encoder from `RoBERTa-Base` to `BERT-Base`, and all models are based on `BERT-Base` for a fair comparison. Note that perfect R@Y is not possible when the ground truth contains less than Y searched tables.

self-supervised pre-training and achieves promising results in various table related NLP tasks. We adopt **TaBERT** (Yin et al., 2020), **TABBIE** (Iida et al., 2021), **TUTA** (Wang et al., 2021), **FORTAP** (Cheng et al., 2022), and **TableFormer** (Yang et al., 2022) as baseline encoders to calculate the column unionability score and adopt our Table Scorer module to obtain the table unionability score. Although, there are more baseline models in this category, we just select representative and SOTA ones. The details of baselines are introduced in Appendix B.

4.2 Results and Analysis

Overall Performance. Table 2 shows the average results on five table union search benchmarks. The proposed AutoTUS consistently outperforms all baselines in MAP and R performance by leveraging *column relational representation learning* (with student's T test $p < 0.05$). More specifically, for five benchmarks, compared to the SOTA baseline, AutoTUS on average achieves 5.1% higher MAP and 3.1% higher R across various benchmarks. Compared with BERT-Base, all table pre-trained and table union search baselines could gain MAP and R performance improvements from better learned tabular representations. The table union search baselines can achieve better results than the table pre-trained baselines by optimizing the *column representations* specific to the table union search task. AutoTUS exploit the *column relational representations* and are able to beat all the table union search baselines. In addition, we study the performance changes brought about by using larger base encoders BERT-Large (Devlin et al., 2019) and RoBERTa-Large (Liu et al., 2019).

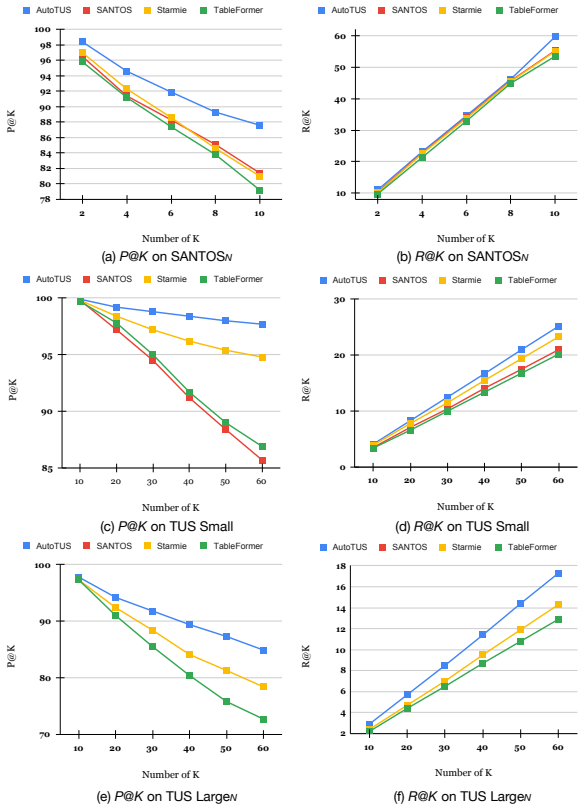
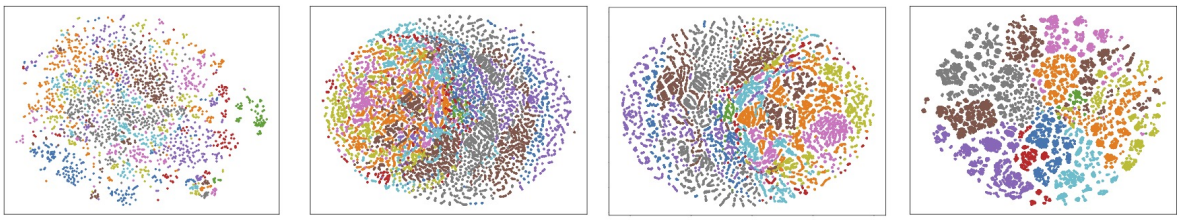


Figure 3: P@Y and R@Y results on different datasets.

From Figure 2, it can be concluded that leveraging a large-scale pre-trained language model with more parameters can lead to better MAP and R, BERT-Large and RoBERTa-Large obtain an average boost of 1.1% and 1.4%, respectively.

Another interesting finding is that, although all the models have decreased performance on the datasets with table noise, the average performance improvement of AutoTUS compared to SOTA is consistent, and even increased to 6.6% on MAP



(a) **AutoTUS** *w/o Column Pair Classification* (b) **AutoTUS** *w/o Automatic Clustering* (c) **AutoTUS** *w/o Real-world Noise Generation* (d) **AutoTUS**

Figure 4: Visualizing column relational representations after t-SNE dimension reduction on TUS large$_N$.

and 3.8% on R. We attribute the improvement of AUTOTUS to the noisy, weak column relational representations from the large pre-trained language models are exploited and refined: we bootstrap the representations obtained from noisy tabular data via self-supervised training schema. More specifically, we give the comparison of P@Y and R@Y as Y changes in Figure 3. We find that AUTOTUS consistently outperforms all baseline models on both metrics P@Y and R@Y as Y changes.

Ablation Study. We conduct ablation studies to show the effectiveness of different modules of AUTOTUS. A number of variants of AUTOTUS are considered: AUTOTUS *w/o Table Noise Generation* removes added table noise in the training data; AUTOTUS *w/o Column Relation Encoder Updater via Pseudo Label Classification* is AUTOTUS without Column Relation Encoder Updater via Pseudo Label Classification and only uses the Contextualized Column Relation Encoder for Column Relation Encoder Updater via Adaptive Clustering; AUTOTUS *w/o Column Relation Encoder Updater via Adaptive Clustering* replaces the suggested soft-assignment clustering techniques with k-means as a hard-assignment alternative.

The general conclusion from ablation study results shown in Table 2 is that the all three modules contribute positively to improve the performance. More specifically, without table noise in training data, AUTOTUS *w/o Table Noise Generation* gives 2.3% less MAP and R averaged on all datasets, especially the noisy SANTOS$_N$ and TUS large$_N$, the drop reaches 3.5%. If we stop exploiting self-supervised signals for column relational features learning, AUTOTUS *w/o Column Relation Encoder Updater via Pseudo Label Classification* brings 12.8% less MAP and R averaged on all datasets. This huge performance drop fully demonstrates the importance of learning and refining column-pair representations. Column Relation Encoder Updater via Adaptive Clustering gives 3.0% MAP and R boost in average when comparing with the hard-assignment alternative (AUTOTUS *w/o Column Relation Encoder Updater via Adaptive Clustering*).

Visualizing Contextualized Column Relational Embeddings. To intuitively show how clustering-enhanced self-supervised training can exploit self-supervised signals to obtain better contextualized column relational representations, we visualize the dimensionally reduced column relational representation space $\mathbb{R}^{2 \cdot h_R}$ using t-SNE (Maaten and Hinton, 2008). We randomly choose 10 base tables from TUS large$_N$ dataset and sample all column relations that belong to the corresponding base table. We show the visualization results in Figure 4 with each column relation being colored according to their ground-truth.

From Figure 4, we can see that AUTOTUS *w/o Column Relation Encoder Updater via Pseudo Label Classification* can assign meaningful semantics to column relations from different base tables, but these unrefined features cannot be tailored for the table union search task. When Column Relation Encoder Updater via Adaptive Clustering is not applied and only k-means are used, AUTOTUS *w/o Column Relation Encoder Updater via Adaptive Clustering* performs a hard assignment of features, and the clustering results appear messy since there are no cluster centroids with high confidence. AUTOTUS *w/o Table Noise Generation* demonstrates that the clustered features is not tolerate to the noise samples, resulting an unclear cluster boundaries. AUTOTUS shows denser and well-separated clusters, which verifies its powerful column relational representation learning ability.

Parameter Analysis: when K is unknown. The Column Relation Encoder Updater via Adaptive Clustering provide the flexibility to explore column relational features without knowing any prior information about the number of clusters.This property is attractive when the number of target clusters is agnostic. We vary K from 10 to 200 and report the

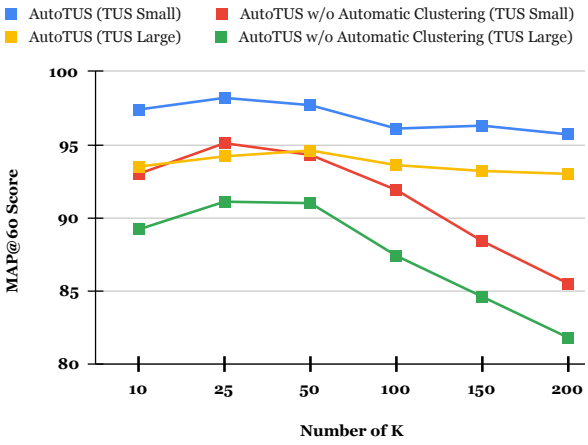


Figure 5: MAP@60 Score with different K.

Table Noise			SANTOS$_N$		TUS large$_N$	
Column Name		Column Data	MAP@10	R@10	MAP@60	R@60
✗		✗	83.0	56.4	80.7	14.9
✗		✓	85.5	57.6	82.1	15.8
✓		✗	86.9	59.2	83.8	16.4
✓	or	✓	87.6	59.7	**84.9**	**17.3**
✓	and	✓	**88.0**	**60.1**	84.7	17.1

Table 3: Table Noise Analysis.

MAP@60 score in Figure 5, the best result is obtained when $K = 25$ for TUS Small and $K = 50$ for TUS Large, slightly larger than the number of base tables in the two datasets, indicating that AUTOTUS actually exploits the number of target clusters as useful prior knowledge. Thanks to the self-supervised training for column relational features exploitation and the flexibility brought by soft-assignment clustering, when we vary K from 10 to 200, AUTOTUS gets a more stable MAP@60 score than AUTOTUS *w/o Column Relation Encoder Updater via Adaptive Clustering*.

Impact of Table Noise. We investigate the effect of table noise in column value and column name. As shown in Section 3.2, we randomly select a corresponding noise from two types of noises. From Table 3, we observe that adding table noise to both column value and column name help to improve the performance, whereas changing column name has a slightly larger effect (2.8% vs. 1.5%). This improvements may related to the fact that column name summarizes the entire column semantically, so adding noise to the column names can make the model more robust to table noise and thus obtain more performance gains. However, adding more noise did not consistently improve performance, suggesting that the model needs to find a trade-off between noisy data perturbation and increased robustness to noise.

5 Related Work

5.1 Table Union Search

In dataset discovery, it is crucial to find related tables in data lakes. The table union search task has recently received considerable attentions (Ling et al., 2013; Lehmberg and Bizer, 2017; Khatiwada and Fan, 2023). The initial attempt is made by Nargesian et al. (2018). The D^{3}L (Bogatu et al., 2020) categories columns into groups by column features. Starmie (Fan et al., 2022) obtains the column features by utilizing a contextualized pre-trained language models. SANTOS (Khatiwada and Fan, 2023) leverages a knowledge base to discover the unionable relations between two tables. More recently, a number of table pre-training models have be developed to obtain the contextualized table representation (Yin et al., 2020; Gong et al., 2020; Dong et al., 2022). However, these methods are not tailored to the table union search task.

5.2 Self-supervised Learning in NLP

Self-supervised learning (SSL) is a method for building models where the output labels are already included in the input data, eliminating the need for additional labeled data (Liu et al., 2021; Hu et al., 2021a,b; Liu et al., 2022d,c). SSL has been widely used in NLP domains such as sentence generation (West et al., 2019; Yan et al., 2021), document processing (You et al., 2021; Ginzburg et al., 2021), natural language inference (Li et al., 2022, 2023), and text reasoning (Klein and Nabi, 2020; Fu et al., 2020; Chen et al., 2022). BERT (Devlin et al., 2019) is one of the most eminent SSL methods which exploit self-supervisions from corpus with next sentence prediction and masked language modeling tasks. In our work, we adopt SSL method to exploit and refine self-supervised signals from tabular data.

6 Conclusions

In this paper, we propose a multi-stage self-supervised table union search framework, which represents column relation as column relational representation and learn this representation in a multi-stage manner that can better describe column relation for unionability prediction. In additional, a table noise generator is proposed to improved the robustness of our approach against the table noise. The conducted extensive experiments demonstrate the effectiveness of the proposed approach.

7 Limitations

We would like to claim our limitations from two perspectives: technical-wise and application-wise.

Technical-wise: We currently only experiment with BERT-Base, BERT-Large, and RoBERTa-Large as the basic encoders. For larger language models, due to limited resources, we have not implemented them.

Application-wise: The experimental data comes from the Open Data repository released by governments of various countries. Although many domains are covered, some domain-specific data, such as biomedical, have not been considered. Furthermore, our tabular data are all from English, open data research in other languages can be considered as a future research direction.

8 Acknowledgement

We thank the reviewers for their valuable comments. Lijie Wen is the corresponding author. Xuming Hu and Lijie Wen were partially supported by the National Key Research and Development Program of China (No. 2019YFB1704003), the National Nature Science Foundation of China (No. 62021002), Tsinghua BNRist and Beijing Key Laboratory of Industrial Bigdata System and Application. Philip S. Yu was partially supported by the NSF under grants III-1763325, III-1909323, III-2106758, SaTC-1930941.

References

Rami Aly, Zhijiang Guo, Michael Sejr Schlichtkrull, James Thorne, Andreas Vlachos, Christos Christodoulopoulos, Oana Cocarascu, and Arpit Mittal. 2021. Feverous: Fact extraction and verification over unstructured and structured information. In *Proc. of NeurIPS: Datasets and Benchmarks Track*.

Alex Bogatu, Alvaro AA Fernandes, Norman W Paton, and Nikolaos Konstantinou. 2020. Dataset discovery in data lakes. In *Proc. of ICDE*, pages 709–720. IEEE.

Dan Brickley, Matthew Burgess, and Natasha Noy. 2019. Google dataset search: Building a search engine for datasets in an open web ecosystem. In *Proc. of WWW*, pages 1365–1375.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. 2020. Hybridqa: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1026–1036.

Yubo Chen, Yunqi Zhang, and Yongfeng Huang. 2022. Learning reasoning patterns for relational triple extraction with mutual generation of text and graph. In *Findings of ACL*, pages 1638–1647.

Zhoujun Cheng, Haoyu Dong, Ran Jia, Pengfei Wu, Shi Han, Fan Cheng, and Dongmei Zhang. 2022. Fortap: Using formulas for numerical-reasoning-aware table pretraining. In *Proc. of ACL*, pages 1150–1166.

Nadiia Chepurko, Ryan Marcus, Emanuel Zgraggen, Raul Castro Fernandez, Tim Kraska, and David Karger. 2020. Arda: automatic relational data augmentation for machine learning. *Proc. of VLDB*, 13(9):1373–1387.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proc. of NAACL-HLT*, pages 4171–4186.

Haoyu Dong, Zhoujun Cheng, Xinyi He, Mengyu Zhou, Anda Zhou, Fan Zhou, Ao Liu, Shi Han, and Dongmei Zhang. 2022. Table pretraining: A survey on model architectures, pretraining objectives, and downstream tasks. *arXiv preprint arXiv:2201.09745*.

Grace Fan, Jin Wang, Yuliang Li, Dan Zhang, and Renée Miller. 2022. Semantics-aware dataset discovery from data lakes with contextualized column-based representation learning. *arXiv preprint arXiv:2210.01922*.

Tsu-Jui Fu, Xin Wang, Scott Grafton, Miguel Eckstein, and William Yang Wang. 2020. Sscr: Iterative language-based image editing via self-supervised counterfactual reasoning. In *Proc. of EMNLP*, pages 4413–4422.

Sainyam Galhotra and Udayan Khurana. 2020. Semantic search over structured data. In *Proc. of CIKM*, pages 3381–3384.

Dvir Ginzburg, Itzik Malkiel, Oren Barkan, Avi Caciularu, and Noam Koenigstein. 2021. Self-supervised document similarity ranking via contextualized language models and hierarchical inference. In *Findings of ACL-IJCNLP*, pages 3088–3098.

Heng Gong, Yawei Sun, Xiaocheng Feng, Bing Qin, Wei Bi, Xiaojiang Liu, and Ting Liu. 2020. Tablegpt: Few-shot table-to-text generation with table structure reconstruction and content matching. In *Proc. of COLING*, pages 1978–1988.

Zhijiang Guo, Michael Schlichtkrull, and Andreas Vlachos. 2022. A survey on automated fact-checking. *Transactions of the Association for Computational Linguistics*, 10:178–206.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Mueller, Francesco Piccinno, and Julian Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. In *Proc. of ACL*, pages 4320–4333.

Xuming Hu, Lijie Wen, Yusong Xu, Chenwei Zhang, and Philip S. Yu. 2020. Selfore: Self-supervised relational feature learning for open relation extraction. In *Proc. of EMNLP*, pages 3673–3682.

Xuming Hu, Chenwei Zhang, Fukun Ma, Chenyao Liu, Lijie Wen, and Philip S. Yu. 2021a. Semi-supervised relation extraction via incremental meta self-training. In *Findings of EMNLP*, pages 487–496.

Xuming Hu, Chenwei Zhang, Yawen Yang, Xiaohe Li, Li Lin, Lijie Wen, and Philip S. Yu. 2021b. Gradient imitation reinforcement learning for low resource relation extraction. In *Proc. of EMNLP*, pages 2737–2746.

Hiroshi Iida, Dung Thai, Varun Manjunatha, and Mohit Iyyer. 2021. Tabbie: Pretrained representations of tabular data. In *Proc. of NAACL-HLT*, pages 3446–3456.

Aamod Khatiwada and Grace Fan. 2023. Santos: Relationship-based semantic table union search. In *Proc. of SIGMOD*.

Tassilo Klein and Moin Nabi. 2020. Contrastive self-supervised learning for commonsense reasoning. In *Proc. of ACL*, pages 7517–7523.

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 *Proc. of ICDE*, pages 468–479. IEEE.

Oliver Lehmberg and Christian Bizer. 2017. Stitching web tables for improving matching quality. *Proc. of VLDB*, 10(11):1502–1513.

Shu'ang Li, Xuming Hu, Li Lin, Aiwei Liu, Lijie Wen, and Philip S. Yu. 2023. A multi-level supervised contrastive learning framework for low-resource natural language inference. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:1771–1783.

Shu'ang Li, Xuming Hu, Li Lin, and Lijie Wen. 2022. Pair-level supervised contrastive learning for natural language inference. *arXiv preprint arXiv:2201.10927*.

Xiao Ling, Alon Y Halevy, Fei Wu, and Cong Yu. 2013. Synthesizing union tables from the web. In *Proc. of IJCAI*.

Aiwei Liu, Xuming Hu, Li Lin, and Lijie Wen. 2022a. Semantic enhanced text-to-sql parsing via iteratively learning schema linking graph. In *Proc. of KDD*, pages 1021–1030.

Aiwei Liu, Xuming Hu, Lijie Wen, and Philip S Yu. 2023. A comprehensive evaluation of chatgpt's zero-shot text-to-sql capability. *arXiv preprint arXiv:2303.13547*.

Ruixue Liu, Shaozu Yuan, Aijun Dai, Lei Shen, Tiangang Zhu, Meng Chen, and Xiaodong He. 2022b. Few-shot table understanding: A benchmark dataset and pre-training baseline. In *Proc. of COLING*, pages 3741–3752.

Shuliang Liu, Xuming Hu, Chenwei Zhang, Shu'ang Li, Lijie Wen, and Philip S. Yu. 2022c. Hiure: Hierarchical exemplar contrastive learning for unsupervised relation extraction. In *Proc. of NAACL-HLT*, pages 5970–5980.

Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. 2021. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, and Philip Yu. 2022d. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Fatemeh Nargesian, Erkang Zhu, Ken Q Pu, and Renée J Miller. 2018. Table union search on open data. *Proc. of VLDB*, 11(7):813–825.

Aécio Santos, Aline Bessa, Christopher Musco, and Juliana Freire. 2022. A sketch-based index for correlated dataset search. In *Proc. of ICDE*, pages 2928–2941. IEEE.

Livio Baldini Soares, Nicholas Fitzgerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *Proc. of ACL*, pages 2895–2905.

Zhiruo Wang, Haoyu Dong, Ran Jia, Jia Li, Zhiyi Fu, Shi Han, and Dongmei Zhang. 2021. Tuta: tree-based transformers for generally structured table pre-training. In *Proc. of SIGKDD*, pages 1780–1790.

Peter West, Ari Holtzman, Jan Buys, and Yejin Choi. 2019. Bottlesum: Unsupervised and self-supervised sentence summarization using the information bottleneck principle. In *Proc. of EMNLP-IJCNLP*, pages 3752–3761.

Yihai Xi, Ning Wang, Shuang Hao, Yiyi Zhang, and Xinyu Chen. 2023. Popularity sensitive and domain-aware summarization for web tables. *Information Sciences*, 621:729–748.

Junyuan Xie, Ross Girshick, and Ali Farhadi. 2016. Unsupervised deep embedding for clustering analysis. In *Proc. of ICML*, pages 478–487.

Yuanmeng Yan, Rumei Li, Sirui Wang, Fuzheng Zhang, Wei Wu, and Weiran Xu. 2021. Consert: A contrastive framework for self-supervised sentence representation transfer. In *Proc. of ACL-IJCNLP*, pages 5065–5075.

Jingfeng Yang, Aditya Gupta, Shyam Upadhyay, Luheng He, Rahul Goel, and Shachi Paul. 2022. Tableformer: Robust transformer modeling for table-text encoding. In *Proc. of ACL*, pages 528–537.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. Tabert: Pretraining for joint understanding of textual and tabular data. In *Proc. of ACL*, pages 8413–8426.

Chenyu You, Nuo Chen, and Yuexian Zou. 2021. Self-supervised contrastive cross-modality representation learning for spoken question answering. In *Findings of EMNLP*, pages 28–39.

Dan Zhang, Madelon Hulsebos, Yoshihiko Suhara, Çağatay Demiralp, Jinfeng Li, and Wang-Chiew Tan. 2020. Sato: contextual semantic type detection in tables. *Proceedings of the VLDB Endowment*, 13(12):1835–1848.

Shuo Zhang and Krisztian Balog. 2017. Entitables: Smart assistance for entity-focused tables. In *Proc. of SIGIR*, pages 255–264.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. Tat-qa: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proc. of ACL-IJCNLP*, pages 3277–3287.

A Dataset Construction Details

For test data, the SANTOS Small benchmark (Khatiwada and Fan, 2023) consists of 550 data lake tables, which are generated from 296 open datasets from Canada, the United Kingdom, the United States, and Australia. The SANTOS Small also has 50 query tables. There are two benchmarks accessible from Nargesian et al. (2018): TUS Small and TUS Large. The TUS Small consists of 1,530 data lake tables that are generated from 10 base table of Canada open data. The TUS Small also has 150 query tables. The TUS Large consists of 5,043 data lake tables that are generated from 32 base table of Canada open data. The TUS Large also has 100 query tables. The SANTOS[4] and TUS[5] benchmarks and their unionable table ground truth are publicly available. For training data, we randomly generated the same number of unlabeled training tables as the labeled tables from base tables of Open Data. Note that the public code for generating tables is publicly available[6] (Koutras et al., 2021; Khatiwada and Fan, 2023). Since table union search task is an unsupervised task, no dev data is obtained.

B The Introduction of Baseline Models

We compare AUTOTUS with two categories of baseline models. The first category of models leverage unlabeled tables for self-supervised pretraining and achieve promising results in the table understanding tasks as baseline encoders to calculate the column unionability score:

(1) **TaBERT** (Yin et al., 2020) is a pretrained language model that simultaneously learns representations for (semi-)structured tables and natural language phrases. 26 million tables and their English contexts make up the vast corpus on which TaBERT was trained.

(2) **TABBIE** (Iida et al., 2021) develops a straightforward pretraining target (corrupt cell identification) that only learns from tabular data and achieves the state-of-the-art on the table-based tasks. TABBIE offers embeddings of all table substructures (cells, rows, and columns), unlike competing techniques, and it also takes far less computing power to train.

(3) **TUTA** (Wang et al., 2021) is a unified pretraining architecture for comprehending typically arranged tables. TUTA improves transformers with three structure-aware techniques after realizing that understanding a table necessitates spatial, hierarchical, and semantic information.

(4) **FORTAP** (Cheng et al., 2022) explores to leverage rhe spreadsheet formulas for table pretraining. FORTAP adopts two self-supervised pretraining objectives, which are derived from formulas, numerical reference prediction and numerical calculation prediction.

(5) **TableFormer** (Yang et al., 2022) is a structurally conscious table-text encoding architecture in which learnable attention biases are used to fully include tabular structural biases.

Of course, there are more baseline models in this category, we just select representative and SOTA models. The second category of models adopt various column representations to calculate the column unionability score:

(6) **D^{3}L** (Bogatu et al., 2020) creates hash-based indexes using the features of the items in a dataset and maps those features to a uniform distance space.

(7) **SATO** (Zhang et al., 2020) is a hybrid machine learning model that uses both the context of the table and the values of the columns to automatically identify the semantic categories of columns in tables.

(8) **Starmie** (Fan et al., 2022) obtains the semantic information included inside tables by utilizing a contrastive multi-column pre-training technique.

(9) **SANTOS** (Khatiwada and Fan, 2023) suggests a definition of unionability that takes connections between columns and their semantics into principled consideration.

[4] https://github.com/northeastern-datalab/santos

[5] https://github.com/RJMillerLab/table-union-search-benchmark

[6] https://delftdata.github.io/valentine/

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section 7

☑ A2. Did you discuss any potential risks of your work?
Section 7

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Abstract and Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 3 and Section 4

☑ B1. Did you cite the creators of artifacts you used?
Section 3 and Section 4

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Section 3 and Section 4

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Section 3 and Section 4

☑ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Section 3, Section 4, and Appendix A

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 4, and Appendix A

C ☑ Did you run computational experiments?

Section 4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 4

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4 and Appendix A

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 4

☐ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Not applicable. Left blank.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.