Structural Deep Encoding for Table Question Answering

Raphaël Mouravieff
Sorbonne Université,
CNRS, ISIR,
F-75005 Paris, France
raphael.mouravieff
@isir.upmc.fr

Benjamin Piwowarski
Sorbonne Université,
CNRS, ISIR,
F-75005 Paris, France
benjamin.piwowarski
@isir.upmc.fr

Sylvain Lamprier
LERIA,
Université d'Angers,
France
sylvain.lamprier
@univ-angers.fr

Abstract

Although Transformers-based architectures excel at processing textual information, their naive adaptation for tabular data often involves flattening the table structure. This simplification can lead to the loss of essential interdependencies between rows, columns, and cells, while also posing scalability challenges for large tables. To address these issues, prior works have explored special tokens, structured embeddings, and sparse attention patterns. In this paper, we conduct a comprehensive analysis of tabular encoding techniques, which highlights the crucial role of attention sparsity in preserving structural information of tables. We also introduce a set of novel sparse attention mask designs for tabular data, that not only enhance computational efficiency but also preserve structural integrity, leading to better overall performance.

1 Introduction

Tabular data is a common data format (Cafarella et al., 2008), with many downstream tasks such as table question answering (Nan et al., 2022) or table fact verification (Chen et al., 2019). Tables present unique challenges to the research community due to their structured format in rows and columns, the heterogeneous nature of the data inside each cell, as well as the vast diversity of tables a model can encounter. In the last few years, transformer models have become the dominant approach for modeling this format.

However, most works flatten a table into a sequence of tokens treating it as a linear text (Lu et al., 2024). This approach introduces critical limitations for the preservation of crucial structural features. Additionally, the quadratic complexity of the self-attention mechanism in transformers makes processing large tables computationally expensive (Tay et al., 2022). Furthermore, it has been shown (Xie et al., 2022) that a variety of these approaches suffer from over-fitting issues.

Several approaches have been proposed to address these challenges. One approach consists in introducing special tokens to explicitly mark rows and columns, as proposed in models like TAPEX and OmniTab (Liu et al., 2021; Jiang et al., 2022). Other approaches propose to capture structural relationships between table elements by using structural embeddings (Herzig et al., 2020; Wang et al., 2021) or biasing attention (Yang et al., 2022).

Despite these advancements, limited research has explored how incorporating prior structural information affects the generalization performance of table processing models. We argue that sparse attention mechanisms, which have been used to manage table size in models like MATE (Eisenschlos et al., 2021), offer an untapped opportunity to leverage structural information. While MATE primarily focuses on managing table size, we propose extending sparse attention patterns to encode structural relationships within tables, enhancing both scalability and generalization capabilities.

In this paper, we systematically evaluate combinations of existing methods for preserving table structure and introduce new **sparse attention masks** specifically tailored to tabular data, as well as new modules designed to retain structural information. Our contributions are threefold:

- **Comprehensive Evaluation**: We systematically assess all existing table encoding techniques from the literature, as well as our newly introduced methods, across multiple dimensions of generalization.

- **Structural Encoding Guidelines**: Based on our findings, we provide practical recommendations for encoding table structure in transformer-based models. Specifically, we show that 1.) Incorporating at least one form of absolute structural encoding enables models to learn table-specific rules, leading to

Findings of the Association for Computational Linguistics: ACL 2025, pages 2389–2402
July 27 - August 1, 2025

improved generalization. 2.) Sparse attention masks, which selectively allow attention between specific table cells, significantly enhance generalization performance.

- **A Novel Sparse Attention Mask for Enhanced Efficiency**: We demonstrate that our proposed sparse attention masks, when integrated with optimized self-attention mechanisms, achieve significant computational speedups. These masks play a crucial role in scaling transformer-based models to efficiently handle large tables.

2 Related Work

2.1 Table models limitations

Current table processing models exhibit several key limitations that make them challenging to deploy in real-world applications.

Generalization Issue: First, table models do not generalize well. As noted in (Xie et al., 2022), table question-answering are easily perturbed by simple structural modifications, such as row or column permutations. In response to these challenges, new datasets focusing on robustness have emerged (Zhou et al., 2024; Zhao et al., 2023). These datasets introduce other perturbations to table structures, further exposing the severe generalization limitations of state-of-the-art models, but do not explain the causes.

Table Size Issue: The quadratic complexity of the self-attention mechanism in transformers poses a significant limitation when dealing with large inputs (Tay et al., 2022; Ye et al., 2023). Models struggle to process large tables efficiently and tables must be truncated, leading to sub-optimal performance. (Ye et al., 2023) further emphasizes these challenges, showcasing the difficulties transformers face when handling extensive table data. Research from (Patnaik et al., 2024) proposes addressing this problem by weighting relevant parts of the table related to the answer, but this solution requires a complex pipeline with two interacting models. Another solution, sparse attention, as proposed in MATE (Eisenschlos et al., 2021), allows handling of larger tables. Their work however did not explore many sparsity patterns and their interaction with other factors.

LLM Problems: The above observations can be extended to Large Language Models (LLMs). (Sui et al., 2024) demonstrates that even billion-parameter LLMs struggle with simple tasks, such as counting rows in large tables. Additionally, findings from (Liu et al., 2024) show that structural variations in tables containing the same content can lead to a significant drop in model performance, particularly in symbolic reasoning tasks. Our approach can assist recent advances in LLMs, which, although typically based on decoder-only architectures, are beginning to integrate a table-specific encoder (Zha et al., 2023).

2.2 Leveraging table structure

Numerous methods have been developed to preserve the inherent structure of tables. The predominant approach in the literature involves linearizing and concatenating the table with the query, while attempting to retain its structural information (Dong et al., 2022).

Input Token Structure: A widely used method introduces special tokens to signal the table structure (Liu et al., 2021; Jiang et al., 2022; Lin et al., 2020). The idea behind this approach is to preserve the table's structural information by adding special tokens to delimit columns, rows, and cells. These tokens allow the model to infer which cells correspond to specific rows or columns.

Structured Embeddings: Another approach involves incorporating specific segment embeddings to help models access and manipulate table structure information. (Shi et al., 2022; Herzig et al., 2020; Wang et al., 2021) propose to use column and row-specific embeddings to encode the position of each token within the table. (Wang et al., 2021) extends this approach with tree-based embeddings to capture hierarchical relationships within the table's structure, including multiple headers.

Bias Attention: (Zayats et al., 2021; Yang et al., 2022) propose biasing the attention mechanism to incorporate prior knowledge about table structure directly into the model's attention layers.

Sparse Attention: The use of attention masks, that consider the structure of tables, has been proposed in (Eisenschlos et al., 2021). Masking attention between tokens of different columns or rows allows to integrate table structure by architecture design. This was however introduced in (Eisenschlos et al., 2021) for computational efficiency purposes only. In this paper, we break down all the methods discussed in this section into individual components and systematically assess their interactions.

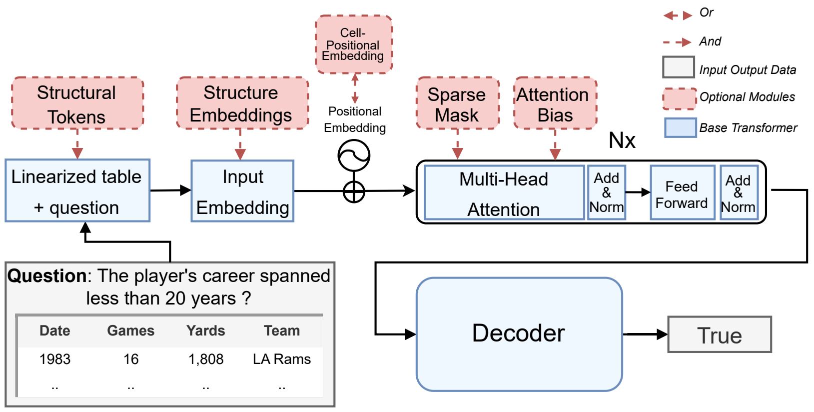

Figure 1: Overview of the encoding pipeline (blue) along with the different steps where table-specific information can be injected (red)

2.3 Generalization experiments

The generalization capabilities of transformers have been a topic of sustained interest within the research community (Hupkes et al., 2023). We list below the different challenges.

Structural Generalization examines how well a model adapts when changing its input structure. Previous works, such as (Saparina and Lapata, 2024), have focused on how small variations affect performance. However, few studies have extended this to larger tables as we do in this work.

Compositional Generalization refers to the ability to recombine known elements to handle new inputs. While there has been substantial work in logical problem domains (Dziri et al., 2024; Zhang et al., 2022), how to handle data tables remains underexplored. To the extent of our knowledge, the only work on tables regarding compositional generalization is that of (Rai et al., 2023), who introduces improved tokenization strategies for tables.

Robustness of Generalization: Robustness, particularly in response to adversarial perturbations, is a critical area of study in table question-answering. (Zhao et al., 2023; Chang et al., 2023) have developed datasets to test specific robustness properties. In this work, we extend these efforts by systematically analyzing model sensitivity to cell repetition.

3 Models and Structural Encoding Components

In this section, we first present our backbone models before specifying various independent methods to encode data tables (Figure 1).

3.1 Backbone

We adopt BART (Lewis, 2019) as the baseline model for our experiments. Following (Liu et al., 2021), (Jiang et al., 2022) who proved that BART can obtain very strong results on table benchmarks. We follow standard table linearization strategies, by concatenating rows sequentially, with the query pre-pended. To the best of our knowledge, this remains the only linearization approach proposed in the literature. Besides the methods described below, we modify BART using a Segment Embedding (Devlin, 2018) that distinguishes the question from the rest of the sequence (i.e., the table). For an illustrative example, refer to Figure 9 (in appendix). All models share this common structure ensuring a fair and consistent comparison across models.

3.2 Special Tokens (T)

Special tokens can be introduced in the linearization to encode structural information in the input sequence. We experiment with three types: T0 (no tokens), T1 (Row Indexed - Cell Tokens), and a novel variant T2 (Row - Column - Cell Tokens). T1, used in models like TAPEX (Liu et al., 2021), marks each start of row with `[ROW n]`, where n is the row number, and each new cell by a cell separator `[CELL]`. T2 is proposed to explore the need for absolute markers of row numbers, which can hinder generalization abilities regarding permutation invariance. It also introduces a column special token `[COL]`, which can be leveraged by structural masks described below.

3.3 Structural Embeddings (E)

An alternative to using special tokens is to add structural Embeddings to different segments of the input. In this paper, we compare E0 (No Structural Embeddings) with E1 (Row Column Embeddings), which corresponds to the Structural Embeddings introduced in TAPAS (Herzig et al., 2020). In the latter, each token embedding X_i is augmented with structural ones, such as: $\tilde{X}i = X_i + E_{\text{row}}(r_i) + E_{\text{col}}(c_i)$, where E_{row} and E_{col} are learnable embeddings specific to the row and column indices, allowing the model to differentiate between cells based on their position.

3.4 Cell/Table Positional Embedding (PE)

We also explore two distinct approaches for positional embedding:
Table Positional Embedding (TPE): the standard approach used in transformers like BART (Lewis, 2019), applying a global sequence of positional embeddings across the entire input sequence.
Cell Positional Embedding (CPE): is a table-specific method where the positional embeddings index resets after each cell (Eisenschlos et al., 2021; Yang et al., 2022)

Note that throughout this paper, we consistently distinguish between **absolute methods**, which assign fixed positional values to tokens based on their location in the input (TPE, E1, T1) and **relative methods**, which remain independent of row and column order (the rest).

3.5 Bias Attention (B)

We use a set of learnable biases B, added to classical transformer's attention scores (before the softmax operation), to incorporate relational information from the table in contextual encoding. We use the code from TableFormer (Yang et al., 2022) for the bias creation, where the attention bias $B_{i,j}$ for the attention of i onto j depends on the relationship between tokens i and j (i.e. cell to column header, etc.). More details about attention biases are given in appendix (sectionB.1). We note B1 for the presence of Bias and B0 the absence.

3.6 Sparse Masking in Attention (M)

We also experiment the use of sparse attention masks to restrict transformer's attention between components of the table, depending on its structure. This is done by adding a mask M to all attention scores, before the softmax function, with $M_{i,j} = -\infty$ if the attention between i and j is masked. We propose six different sparse masks, ranging from M1 (the least sparse, corresponding to MATE) to M6 (the most sparse), with M0 representing the case where no sparse attention is applied. Some of these sparse masks necessitate the structural tokens T2 (section 3.2). These masks are marked with * to differentiate them from the others. Full details of our mask schemes can be found in Table 1, and a more visual example is provided in appendix (Figure 8).

Attention	M1	M2	M3	M4*	M5*	M6*
$q_i \leftrightarrow w_{r,c,k}$	X	X	X	X	X	X
$q_i \leftrightarrow q_{i'}$	X	X	X	X	X	X
$w_{r,c,k} \leftrightarrow w_{r',c,k'}$	X	X		X		
$w_{r,c,k} \leftrightarrow w_{r,c',k'}$	X		X		X	
$w_r \leftrightarrow w_{r,c,k}$				X	X	X
$w_c \leftrightarrow w_{r,c,k}$				X	X	X
$w_{r,c} \leftrightarrow w_{r,c,k}$				X	X	X
$w_T \leftrightarrow w_{r,c,k}$				X	X	X

Table 1: Summary of allowed attention ($\leftrightarrow$) patterns for different sparse masks. We denote q_i a query token, $w_{r,c,k}$ a token at position k in the cell located row r / column c. Then, $w_{r,c,k} \leftrightarrow w_{r',c,k'}$ indicates that attention is *not masked* between tokens within the same column across different rows. Additionally, for T2, w_c, w_r, $w_{r,c}$ and w_T denote the column c, row r, cell (r, c) and table tokens respectively.

4 Experimental Setup

In addition to the real-world datasets introduced later, we use synthetic data for a more fine-grained impact analysis of the encoding factors. These synthetic datasets are designed to assess the generalization capabilities of the models by evaluating their robustness to missing values, structural changes, compositional generalization of queries, and resilience to correlation between cell contents often found in real world datasets (referred to as mixability in the following).

4.1 Synthetic Data Generation

A dataset $\mathcal{T}$ corresponds to a set of triplets (T, Q, A), where T is a table, q is an SQL query, and A is the target answer.
Table Generation: Each table T is a matrix, N_{row}^T rows and N_{col}^T columns, both uniformly sampled from $\mathcal{U}(6, 8)$ for each sample. Each cell (r, c) in

T is a sequence of tokens $w_{r,c,k}$, resulting from the tokenization of a random integer, sampled from $\mathcal{U}(V)$ with $V = \{0, 1, \ldots, 999\}$.

Query Generation: SQL queries are based on 10 common patterns from SQUALL (Shi et al., 2020), focusing on simple selection tasks, that are instantiated according to contents of data they are applied on. We use the Fuzzingbook library[1] to generate a set of variations from each SQL query template. For details about the SQL templates used, please refer to the appendix (section B.2). It is worth noting that SQL is a natural choice for studying Table Question Answering due to the structured alignment between text and SQL queries (Wang et al., 2019).

4.2 Disturbances from In-Domain Data

Structural Generalization: To evaluate the model's robustness to structural changes, we test its performance on tables of varying sizes, both larger and smaller than those seen during training. The model is tested on tables with dimensions outside the training range, i.e. $N^T_{row}, N^T_{col} \in \{4, 5, 9, 10, 11, 12\}$.

Consistency Robustness: Token repetitions make the task harder since the model cannot rely on the token semantic. In our experiments on consistency robustness, we select a random word v_0 from our vocabulary $v \in V$. For each cell, we replace its content with v_0 with a probability R. We use either $R = 0.2$ or $R = 0.4$ (with uniform probability).

Compositional Generalization: During training, we use an ensemble of 10 SQL templates (see Section B.2 in appendix). These templates include basic patterns such as `SELECT cx FROM table WHERE cy IN ...` or `SELECT cx FROM table LIMIT k`. To evaluate compositional generalization, we use queries that combine these known components in new ways, such as `SELECT cx FROM table WHERE cy IN ... LIMIT k`. This setup allows us to assess the model's ability to generalize to more complex queries by combining simpler ones.

Mixability Robustness: We use a parameter $S \in [0, 1]$ to control how deterministic the table content generation is. With $S = 1$, a cell content if fully determined by the previous cells in the row, while for $S = 0$, we use a random generation (same table generation process as in the training set). We detail the generation procedure in appendix (Section B.3).

[1] https://github.com/uds-se/fuzzingbook

4.3 Real Datasets and Evaluation

Finally, we use the following real-world datasets in our experiments. **WikiTableQuestions (WTQ)** (Pasupat and Liang, 2015) is a challenging benchmark for table question answering, as it includes numerical reasoning questions, tables with missing values, and noisy columns containing a mix of text and numerical cells. **WikiSQL (WSQL)** (Zhong et al., 2017) is another table-based question answering dataset, where we use the provided SQL supervision to train models, mapping tables and questions to execution results. This dataset is considered to be simpler than WikiTableQuestions, as it more closely follows structured SQL queries.

4.4 Training pipeline

We initialize our models using pre-trained BART weights (Lewis, 2019). Our SQL execution procedure serves as a common technique for intermediate pre-training on tabular data (Liu et al., 2021). For our training procedure on artificial datasets, we run up to 200k steps with an early stopping patience of 15 to ensure convergence, as suggested by (Csordás et al., 2021). We use a batch size of 8, a context length of 512, and a learning rate of 3×10^{-5}. For fine-tuning on real datasets, we adopt the same hyperparameters as described in (Liu et al., 2021).

5 Experiments

We evaluate our models using denotation accuracy (DA) on both real and synthetic datasets. Denotation accuracy measures correctness by comparing ground-truth and predicted outputs, considering a prediction correct if the sets of values match, i.e. irrespective of order.

5.1 ANOVA Decomposition of Structural Factors in Tabular Encoding

We use ANOVA to evaluate the impact of structural components (Table 2).

Main Effects: Positional Embeddings (PE), and Tabular Structure Embeddings (E) significantly affect model performance (and have a strong interaction PE $\times$ E). Positional embeddings exhibit the strongest effect ($\eta^2 \in [0.18, 0.27]$), where TPE consistently outperforms CPE. Tabular structure embeddings also improves accuracy (E1 $>$ E0, $\eta^2 \in [0.15, 0.30]$). In contrast, sparse tokens (T), Bias (B) and mask (M) – when considered independently – exhibit no significant impact on model performance. These results indicate that absolute table encoding methods, such as TPE and RCE, are critical for table-based question answering, as models struggle to generalize with purely relative encodings. We hypothesize that without CPE or TPE, the decoder struggles to locate relevant information.

Interaction Effects: Mask M has a strong effect on performance when combined with specific positional PE or structural embeddings E. For example, TPE consistently outperforms CPE when using masks (e.g., T2 with a mask). Furthermore, there is a notable interaction between PE and E ($\eta^2 \in [0.19, 0.26]$).

Non-significant Factors: Special tokens (T) and Bias (B) do not significantly impact performance, either alone or in interactions (except Bias with Mask; see Appendix C.4). This suggests that token structure encoding is less critical compared to sparsity mechanisms or positional embeddings, aligning with prior work emphasizing the role of better structure-aware representations.

Table 2: ANOVA results showing the effect sizes (η^2) and p-values for each factor on performance across different tasks (In Domain, Structure, Consistency, Compositional, and Mixability). Significant effects are indicated with bold text if ($p \leq 0.05$).

Factor	In Domain	Structure	Consistency	Compositional
	η^2 (p-value)	η^2 (p-value)	η^2 (p-value)	η^2 (p-value)
T	0.00	0.00	0.00	0.00
M	0.04	0.07	0.01	0.00
PE	**0.19**	**0.27**	**0.19**	**0.26**
B	0.01	0.01	0.00	0.00
E	**0.20**	**0.15**	**0.30**	**0.26**
TM	0.00	0.01	0.01	0.02
T×PE	0.00	0.00	0.00	0.01
T×B	0.00	0.00	0.00	0.00
T×E	0.00	0.00	0.00	0.00
M×PE	**0.08**	**0.05**	**0.07**	**0.04**
M×B	0.02	**0.04**	0.02	0.01
M×E	0.09	0.04	0.07	0.04
PE×B	0.01	0.00	0.00	0.00
PE×E	**0.19**	**0.21**	**0.18**	**0.26**
B×E	0.01	0.01	0.01	0.00

5.2 Impact of Structural Encoding: Performance Differences Across Models

We analyze further key ANOVA findings by focusing on the most significant factors (Figure 2). The figure illustrates performance differences between two treatments of a factor (e.g., PE), while keeping other factors (e.g., T, E, M, B) constant. First, sparse masking M1 consistently improves performance over M0. This supports the hypothesis that restricting cell attention in table encoding mitigates spurious correlations and enhances generalization. Second, absolute encoding generally yields better generalization than relative encoding (TPE > CPE and $E1 > E0$). Although counterintuitive, since relative encodings should help capturing table invariance (e.g. swapping two rows or columns), this is likely due to the task nature, where SQL query interactions with table during question answering require absolute embeddings to capture rule-based relationships effectively. We conduct the same experiments for tokens and bias, with results provided in Appendix C.5.

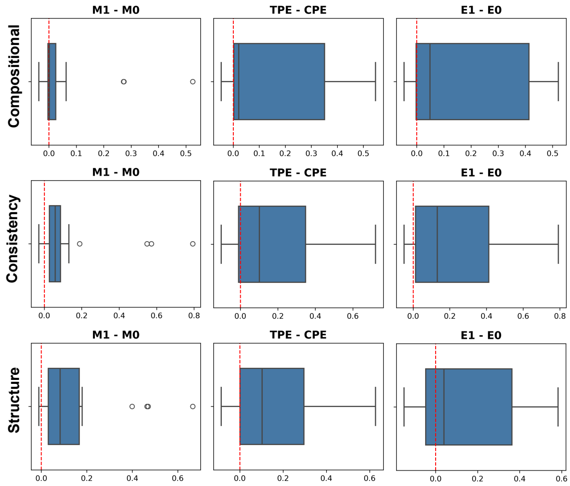

Figure 2: Denotation Accuracy differences between two structural encoding components (left − right) while keeping all other factors unchanged.

5.3 Validating Structural Encoding Insights: Consistency Across Synthetic and Real-World Data

To validate our findings on real-world data, we evaluated the WikiSQL dataset, testing 120 model configurations (see Table 3).

Performance of M1: Overall, the M1 mask outperforms the baseline (M0) across all configurations, demonstrating that introducing a sparse mask significantly enhances generalization. More experiments on M1 can be find in Appendix C.2.

Performance of Other Masks: While M1 offers a good trade-off between sparsity and performance, other masks such as M5* or M3, despite being significantly sparser (see Figure 8 in the appendix), still outperform M0 by a wide margin and achieve comparable performance to M1. Notably, these masks are at least twice as sparse as M1, with M6 having 99% of its attention masked.

Absolute vs. Relative Encoding: Some configurations failed to converge due to overfitting, particularly when absolute information is absent (CPE, E0). Without absolute positional cues, the model struggles to differentiate cells, leading to poor generalization. This aligns with our ANOVA analysis on synthetic data – see Appendix figure C.3 for more details.

5.4 M3: An Efficient Sparse Attention Mask for Faster and Improved Performance

In this section, we focus on M3 which has a very good performance overall, along with a high sparsity level, and study its impact on model efficiency. For this experiment, we use FlexAttention and FlashAttention (FA2) (Dao et al., 2022), both optimized for sparse matrix operations. FlexAttention, a PyTorch module specifically designed for efficient sparse attention computation, performs optimally with block-based sparse attention masks, aligning well with the structure of M3 (see Figure 8 in appendix). The results are presented in Figure 3. The cumulative distribution of WikiTableQuestions (blue curve) reveals that many tables exceed 1024 tokens, the common limit in fine-tuned models, restricting encoding to 80% of WikiTableQuestions. Sparse masks mitigate this limitation. We assessed Forward and Backward Speedup using M3 across sequence lengths of 1024, 2048, 4096, 8192, and 16,384 tokens. Speedup is computed as the ratio of the accelerated masked attention (either Flex or Flash) to the standard PyTorch Sdpa attention. Up to 4096 tokens, FA2 is the most efficient, achieving a 2× speedup over standard attention at 2048 tokens. For longer sequences, FlexAttention reach a 50× forward speedup and (16× for backward) when encoding tables of 16,384 tokens.

Having established M3 computational efficiency, we now evaluate its empirical performance on synthetic and real datasets, comparing against literature baselines and our own implementations (see Table 4). From Table 4, we observe that the M3 mask performs well on the ALL dataset, which represents the average results across all our synthetic datasets, and remains competitive on WikiSQL. This highlights its effectiveness as a candidate for ultra-sparse table encoding. Additionally, compared to existing models, our M3 implementation maintains strong performance while offering significant computational advantages, making it a promising solution for scalable table-based

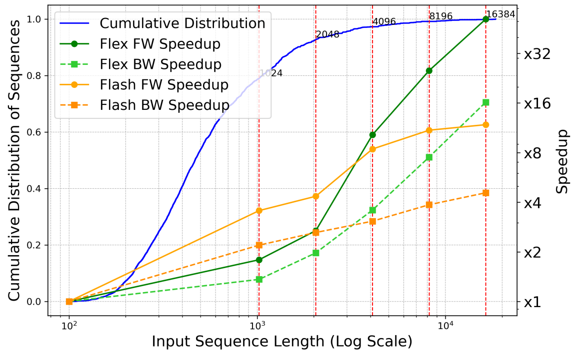

Figure 3: Cumulative Distribution of Sequence Lengths and Relative Computation Speedup: The primary y-axis (left) represents the cumulative distribution of sequence lengths in log scale, while the secondary y-axis (right) shows the relative computation speedup for FlexAttention and FlashAttention2 across different sequence lengths in x-axis.

question-answering tasks.

5.5 Enhancing Generalization with Sparse Attention: The Impact of Non-Random Table Structures

We have identified key factors for encoding tables, and now we explore another critical element for generalization, namely table structure determinism. In datasets from the literature, the number of unique tables is relatively low. For instance, WikiTableQuestions contains only 2k unique tables in a training set of 11k examples. To investigate the impact of deterministic table structures, we compared model performance when trained on either fully deterministic or randomized data.

Figure 4 compares the performance of two models, TAPAS and TAPAS+M1, across varying levels of mixability level (section 4.2). The top row shows TAPAS results, while the bottom row displays TAPAS+M1 results. Both models are trained on data with maximum mixability (S=1) and a restricted number of rows and columns, and tested on datasets with decreasing mixability (0.8 to 0) and larger tables.

We observe that TAPAS exhibits significant overfitting, with performance declining both "In Domain" (outlined in red) and "Out of Domain" (larger tables). In contrast, the TAPAS+M1 model (bottom row) demonstrates resilience to these shifts. Even at lower mixability levels, TAPAS+M1 maintains high and stable accuracy, indicating that the sparse mask (M1) helps mitigate overfitting and

Table 3: This table summarizes all possible model combinations from the literature (T, E, PE, M, B) as well as our proposed configurations (M2, ..., M6, and T2) evaluated on the WikiSQL dataset.

PE	E1	T	M0		M1		M2		M3		M4*		M5*		M6*	
			B0	B1	B0	B1	B0	B1	B0	B1	B0	B1	B0	B1	B0	B1
CPE	E0	T2	26.1	38.6	81.3	81.6	30.2	29.7	58.0	56.3	29.9	29.9	61.4	57.5	29.2	28.8
		T1	26.1	26.6	82.3	81.5	30.0	30.0	57.1	54.7						
		T0	22.7	23.1	79.6	76.8	29.8	29.5	36.5	34.4						
	E1	T2	75.7	73.0	82.1	81.6	78.5	77.6	81.8	81.2	79.2	78.6	81.9	81.2	29.3	29.6
		T1	79.0	79.0	82.2	81.5	78.5	78.0	82.4	81.4						
		T0	29.8	28.8	79.2	78.3	77.2	78.0	71.0	66.2						
TPE	E0	T2	79.4	79.6	82.0	82.0	78.9	78.2	78.7	79.1	80.1	80.0	79.0	78.5	80.5	80.4
		T1	79.5	79.6	81.9	82.4	79.3	78.8	79.2	79.5						
		T0	70.6	71.4	82.5	82.5	79.3	78.9	75.3	73.9						
	E1	T2	79.4	79.3	82.2	82.3	78.6	78.0	79.8	79.4	80.5	80.5	79.7	79.3	80.4	79.8
		T1	79.6	79.7	81.9	81.3	79.0	78.5	79.1	79.9						
		T0	77.4	77.5	82.6	82.6	79.2	79.1	77.2	76.8						

Table 4: Comparison of Model Performance on the Synthetic dataset. "ALL" represents the average score across Structure, Compositionality, and Robustness, alongside WikiSQL test data.

Model	**ALL**	**WSQL**
Literature Models		
MATE (Eisenschlos et al., 2021)	79.2	82.6
TableFormer (Yang et al., 2022)	23.1	60.5
TAPAS (Herzig et al., 2020)	77.4	78.0
TAPEX (Liu et al., 2021)	79.5	74.7
Our models		
T2 M0 TPE B E1	79.3	78.5
T2 M3 TPE B E1	79.4	80.3

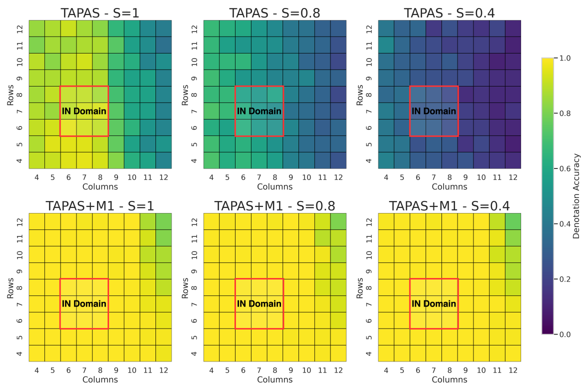

Figure 4: Results for TAPAS and TAPAS+M1 under varying Mixability levels (S). All models have been trained on data with S=1, where the transition matrix for table creation is fully deterministic, and tested on increasingly challenging similarity levels, down to S=0, where the transition matrix is uniformly random. For this experiment, we exclusively used the "SELECT cx WHERE cy = vy" template SQL query.

improves generalization to out-of-domain data.

6 Conclusion and Recommendations

We conducted a comprehensive analysis of encoding techniques for data tables, highlighting the importance of sparse attention and absolute positional encoding in model generalization. Our study systematically evaluated table encoding components, as well as their interactions, providing insights for improving table representation in Transformer-based architectures.

Our findings show that sparse attention masks reduce spurious correlations and enhance structural representation. In particular, we propose the mask M3 that achieves a good efficiency-effectiveness trade-off. Additionally, we show that absolute positional encoding (TPE, E1) is essential, as models relying solely on relative encoding struggle with generalization. Beyond accuracy, sparse masks like M3 achieve up to 50× forward and 16× backward speedup for large tables, enabling efficient scaling for real-world applications.

Overall, our work provides a foundation for efficient and scalable transformers for information extraction from tables of data. High-quality encodings have proven essential when integrating these representations into LLMs, particularly in domains like visually rich document understanding (Ma et al., 2024; Lee et al., 2023). Future works will investigate how to leverage our findings in decoder-only LLM architectures.

7 Limitations

Our study highlights the effectiveness of absolute table encoding for generalization in question-answering tasks. While relative encoding is often expected to enhance generalization, current approaches may not fully capture rule-based relationships in SQL queries. This opens an exciting direction for future work to develop improved relative encoding mechanisms that better integrate structural rules.

Additionally, while our experiments primarily rely on synthetic datasets, we include preliminary evaluations on real-world data (WTQ and WSQL). Expanding real-world benchmarks will further strengthen our findings and enhance applicability across diverse scenarios.

Lastly, our analysis focuses on the BART model as a baseline. Extending this work to other encoder-decoder architectures and decoder-only models presents an exciting opportunity for future research, potentially extending the impact of our approach.

8 Acknowledgments

This work was partly funded by the ACDC (ANR-21-CE23-0007) and GUIDANCE (ANR-23-IAS1-0003) projects. Experiments were performed using HPC resources from GENCI-IDRIS (Grants AD011014032R1, AD011014110, and A0151014638).

References

Michael J Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. Webtables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment*, 1(1):538–549.

Shuaichen Chang, Jun Wang, Mingwen Dong, Lin Pan, Henghui Zhu, Alexander Hanbo Li, Wuwei Lan, Sheng Zhang, Jiarong Jiang, Joseph Lilien, et al. 2023. Dr. spider: A diagnostic evaluation benchmark towards text-to-sql robustness. *arXiv preprint arXiv:2301.08881*.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2019. Tabfact: A large-scale dataset for table-based fact verification. *arXiv preprint arXiv:1909.02164*.

Róbert Csordás, Kazuki Irie, and Jürgen Schmidhuber. 2021. The devil is in the detail: Simple tricks improve systematic generalization of transformers. *arXiv preprint arXiv:2108.12284*.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.

Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, et al. 2024. Faith and fate: Limits of transformers on compositionality. *Advances in Neural Information Processing Systems*, 36.

Julian Martin Eisenschlos, Maharshi Gor, Thomas Müller, and William W Cohen. 2021. Mate: multi-view attention for table transformer efficiency. *arXiv preprint arXiv:2109.04312*.

Jonathan Herzig, Paweł Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Martin Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. *arXiv preprint arXiv:2004.02349*.

Dieuwke Hupkes, Mario Giulianelli, Verna Dankers, Mikel Artetxe, Yanai Elazar, Tiago Pimentel, Christos Christodoulopoulos, Karim Lasri, Naomi Saphra, Arabella Sinclair, et al. 2023. A taxonomy and review of generalization research in nlp. *Nature Machine Intelligence*, 5(10):1161–1174.

Zhengbao Jiang, Yi Mao, Pengcheng He, Graham Neubig, and Weizhu Chen. 2022. Omnitab: Pre-training with natural and synthetic data for few-shot table-based question answering. *arXiv preprint arXiv:2207.03637*.

Kenton Lee, Mandar Joshi, Iulia Turc, Hexiang Hu, Fangyu Liu, Julian Eisenschlos, Urvashi Khandelwal, Peter Shaw, Ming-Wei Chang, and Kristina Toutanova. 2023. Pix2struct: Screenshot parsing as pretraining for visual language understanding. *Preprint*, arXiv:2210.03347.

M Lewis. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2020. Bridging textual and tabular data for cross-domain text-to-sql semantic parsing. *arXiv preprint arXiv:2012.12627*.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2021. Tapex: Table pre-training via learning a neural sql executor. *arXiv preprint arXiv:2107.07653*.

Tianyang Liu, Fei Wang, and Muhao Chen. 2024. Rethinking tabular data understanding with large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 450–482, Mexico City, Mexico. Association for Computational Linguistics.

Weizheng Lu, Jiaming Zhang, Jing Zhang, and Yueguo Chen. 2024. Large language model for table processing: A survey. *arXiv preprint arXiv:2402.05121*.

Feipeng Ma, Yizhou Zhou, Hebei Li, Zilong He, Siying Wu, Fengyun Rao, Yueyi Zhang, and Xiaoyan Sun. 2024. Ee-mllm: A data-efficient and compute-efficient multimodal large language model. *arXiv preprint arXiv:2408.11795*.

Linyong Nan, Chiachun Hsieh, Ziming Mao, Xi Victoria Lin, Neha Verma, Rui Zhang, Wojciech Kryściński, Hailey Schoelkopf, Riley Kong, Xiangru Tang, et al. 2022. Fetaqa: Free-form table question answering. *Transactions of the Association for Computational Linguistics*, 10:35–49.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *arXiv preprint arXiv:1508.00305*.

Sohan Patnaik, Heril Changwal, Milan Aggarwal, Sumita Bhatia, Yaman Kumar, and Balaji Krishnamurthy. 2024. Cabinet: Content relevance based noise reduction for table question answering. *arXiv preprint arXiv:2402.01155*.

Daking Rai, Bailin Wang, Yilun Zhou, and Ziyu Yao. 2023. Improving generalization in language model-based text-to-sql semantic parsing: Two simple semantic boundary-based techniques. *arXiv preprint arXiv:2305.17378*.

Irina Saparina and Mirella Lapata. 2024. Improving generalization in semantic parsing by increasing natural language variation. *arXiv preprint arXiv:2402.08666*.

Peng Shi, Patrick Ng, Feng Nan, Henghui Zhu, Jun Wang, Jiarong Jiang, Alexander Hanbo Li, Rishav Chakravarti, Donald Weidner, Bing Xiang, et al. 2022. Generation-focused table-based intermediate pre-training for free-form question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11312–11320.

Tianze Shi, Chen Zhao, Jordan Boyd-Graber, Hal Daumé III, and Lillian Lee. 2020. On the potential of lexico-logical alignments for semantic parsing to sql queries. *arXiv preprint arXiv:2010.11246*.

Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. 2024. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 645–654.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. Efficient transformers: A survey. *Preprint*, arXiv:2009.06732.

Bailin Wang, Ivan Titov, and Mirella Lapata. 2019. Learning semantic parsers from denotations with latent structured alignments and abstract programs. *arXiv preprint arXiv:1909.04165*.

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 *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1780–1790.

T Wolf. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir Radev, Caiming Xiong, Lingpeng Kong, Rui Zhang, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2022. UnifiedSKG: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 602–631, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jingfeng Yang, Aditya Gupta, Shyam Upadhyay, Luheng He, Rahul Goel, and Shachi Paul. 2022. Tableformer: Robust transformer modeling for table-text encoding. *arXiv preprint arXiv:2203.00274*.

Yunhu Ye, Binyuan Hui, Min Yang, Binhua Li, Fei Huang, and Yongbin Li. 2023. Large language models are versatile decomposers: Decomposing evidence and questions for table-based reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 174–184.

Vicky Zayats, Kristina Toutanova, and Mari Ostendorf. 2021. Representations for question answering from documents with tables and text. *arXiv preprint arXiv:2101.10573*.

Liangyu Zha, Junlin Zhou, Liyao Li, Rui Wang, Qingyi Huang, Saisai Yang, Jing Yuan, Changbao Su, Xiang Li, Aofeng Su, et al. 2023. Tablegpt: Towards unifying tables, nature language and commands into one gpt. *arXiv preprint arXiv:2307.08674*.

Yi Zhang, Arturs Backurs, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, and Tal Wagner. 2022. Unveiling transformers with lego: a synthetic reasoning task. *arXiv preprint arXiv:2206.04301*.

Yilun Zhao, Chen Zhao, Linyong Nan, Zhenting Qi, Wenlin Zhang, Xiangru Tang, Boyu Mi, and Dragomir Radev. 2023. RobuT: A systematic study of table QA robustness against human-annotated adversarial perturbations. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6064–6081, Toronto, Canada. Association for Computational Linguistics.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

Wei Zhou, Mohsen Mesgar, Heike Adel, and Annemarie Friedrich. 2024. FREB-TQA: A fine-grained robustness evaluation benchmark for table question answering. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2479–2497, Mexico City, Mexico. Association for Computational Linguistics.

A Reproductibility statement

All experiments in this paper were conducted using publicly available datasets and open-source libraries, including (Pasupat and Liang, 2015; Shi et al., 2020; Zhong et al., 2017). Detailed descriptions of model architectures, hyperparameters, and training configurations are provided in the section 4.4. To implement our experiments, we used the transformer library (Wolf, 2019). The code will be published upon publication. All experiments were conducted on work stations with 80GB NVIDIA A100, 16 or 32GB NVIDIA V100 SXM2-HBM2 GPUs.

B Methodology Details

B.1 TableFormer Bias

In TABLEFORMER, attention biases are designed to handle various table-text structural relationships, using 13 types of biases to capture row, column, header, and sentence relationships in tabular data. These biases include mechanisms for recognizing same row and column information, linking cells to their respective headers, and incorporating sentence-to-cell grounding to enhance the understanding of tables in context. Each bias type is then associated with a learnable scalar. For more details, see (Yang et al., 2022).

B.2 SQL Query for training.

The following SQL query templates were used to generate synthetic data for our experiments. Each template includes various possible choices for conditions, making them versatile and applicable to different table structures:

Parameter	Values
Input Token Structures	T1
	T2
	T0
Mask Sparsity Levels	M0
	M1
	M2
	M3
	M4
	M5
	M6
Positional Embeddings	TPE
	CPE
Encoding Structure Bias	Bias
	No Bias
Tabular Structure Embeddings	E1
	E0

Table 5: Overview of the experimental parameters used in our study. These include input token structures, mask sparsity levels, positional embeddings, encoding structure biases, and tabular structure embeddings.

Table 6: State-of-the-art models and structural encoding methods.

Model	T	E1	PE	B	M
MATE	T0	E1	CPE	B0	M1
TableFormer	T0	E0	CPE	B0	M0
TAPAS	T0	E1	TPE	B0	M0
TAPEX	T1	E0	TPE	B0	M0

- `SELECT cx WHERE cy =|!= vy AND|OR cz =|!= vz AND|OR cw =|!= vw AND|OR cl =|!= vl`
- `SELECT cx WHERE cy =|!= vy AND|OR cz =|!= vz AND|OR cw =|!= vw`
- `SELECT cx WHERE cy =|!= vy AND|OR cz =|!= vz`
- `SELECT cx WHERE cy =|!= vy`
- `SELECT cx`
- `SELECT cx LIMIT k`, where $k \in \{1, 2, 3\}$
- `SELECT cx WHERE cy = (SELECT cy WHERE cy = vy)`
- `SELECT cx WHERE cy IN (vy | vy, vy | vy, vy, vy)`

B.3 Generating tables for mixability

To generate tables where cell content is more or less determined by the previous cells in the row,

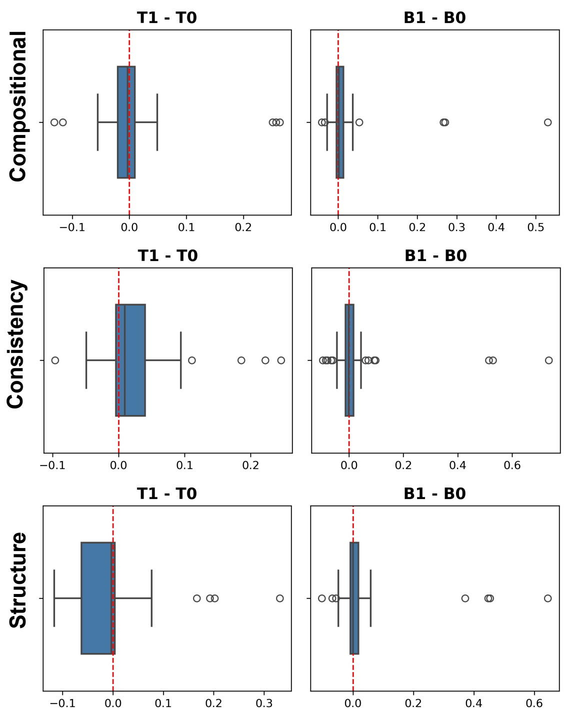

Figure 5: This figure highlights the differences between two structural encoding components while keeping all other factors unchanged.

we use two base transition matrices: a deterministic transition matrix M^{deter}, and a uniform matrix M^{unif}, where transitions are equally probable. The transition matrix for our experiments is a weighted combination of these two matrices:

$$M^{\text{transi}} = S \cdot M^{\text{deter}} + (1 - S) \cdot M^{\text{unif}}$$

During training, we set $S = 1$ for structured tables, while during testing, we use $S = 0$ to introduce randomness, simulating real-world data variability.

C Additional Results

C.1 Performance Differences Across Models

We report in Figure 5 the effect of special tokens (T) and bias (B) on the three synthetic datasets.

C.2 Mask Sparsity Effects

Based on the ANOVA results and the strong significance of mask sparsity on accuracy variation, we selected compositions of factors from the literature to evaluate against the sparse mask M1, across both generated and real-world datasets (WikiSQL, WikiTableQuestions). M1 was tested individually because it had previously demonstrated the best overall performance in evaluations. As shown in Table 7, the results demonstrate that adding the M1 mask leads to substantial performance improvements in both in-domain, out-of-domain generalization tasks and real-world-datasets.

Across all out-domain data (Structure, Robustness, Compositional, Mixability), the addition of the M1 mask yields significant gains, with improvements reaching over 10%. For example, the T0/M1 model outperforms the T0/M0 model by 9.7 points in robustness and 5.6 points in compositional generalization. This indicates that sparse masks help the models generalize better.

For real datasets, the sparse mask again improves performance. The T0/M1 model achieves 46.8% on WikiSQL, compared to 34.6% for the T0/M0 model, representing a significant 12.2-point gain. This underscores the practical benefit of sparse masks in real-world applications, where data distributions can be more variable and noisy.

Overall, this experiment highlights the critical role of mask-based sparsity (M1) in improving model performance across a wide range of tasks and datasets. The results not only confirm the hypothesis that sparse masks enhance generalization but also show that these gains extend across both synthetic and real-world datasets, making them a valuable addition to existing models in the literature.

C.3 Alignment Between Synthetic and Real Data:

Figure 6 assesses consistency across datasets. Red markers indicate perfect agreement, with standardized means comparing encoding effects across synthetic and real data. A strong correlation confirms alignment, except for M6, where extreme sparsity prevents the model from recognizing table vocabulary, degrading performance.

C.4 Bias and Sparse Masks.

As the ANOVA results show (see Table 2), the interaction between bias and mask has a significant effect on structure. To analyze this interaction effect, we average the bias and mask results across interaction configurations, as shown in Table 8. We observe that the models that benefit the most from the addition of bias are those using no mask (M0). For all other masks, adding bias has only a minor effect.

T

Table 7: Comparison of model performance on generalization test sets, highlighting the effect of mask M1 across both in-domain and out-of-domain tasks (Structure (Struc), Mixability (Mixab), Consistency (Consi), Compositional (Comp) and real datasets (WTQ, WSQL). Models with mask M1 generally show improved accuracy, particularly in robustness and compositional generalization, compared to models without the mask (M0). This demonstrates the impact of mask-based sparsity on handling diverse data distributions.

Model	InDomain	Mixab	Consi	Comp	Struct	WSQL	WTQ
T0/M0/TPE/B0/E1$_{\textbf{tapas}}$	99.8	98.9	76.3	62.8	66.3	84.8	34.6
T0/M1/TPE/B0/E1	99.9	98.8	88.4	62.8	76.0	87.6	46.8
T1/M0/TPE/B0/E0$_{\textbf{tapex}}$	98.4	98.9	71.3	62.9	66.3	87.1	52.4
T1/M1/TPE/B0/E0	99.8	99.1	90.2	62.6	71.5	86.0	51.4
T0/M0/TPE/B1/E0$_{\textbf{tableformer}}$	99.9	98.5	80.2	62.7	78.0	83.3	42.4
T0/M1/TPE/B1/E0	99.8	99.2	86.2	62.4	81.1	86.9	46.5

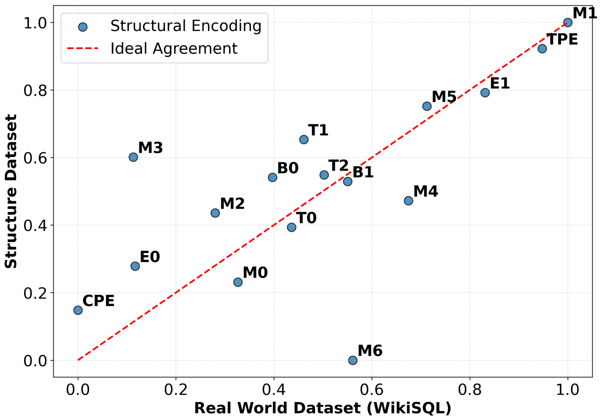

Figure 6: This figure highlights the agreement between our "structure" synthetic dataset (x-axis) and the real-world WikiSQL dataset (y-axis) – with normalized (max) metrics, and using the mean over all experiments with a given table encoding component. The diagonal red line represents the ideal agreement where identical results are obtained on both datasets.

C.5 Impact of Structure Embeddings

We tested the combined effect of structure embedding and positional encoding across multiple datasets, as shown in Figure 7. The results clearly demonstrate that models using CPE without E1 struggle to converge – most probably because there is no more absolute positioning information. However, using standard positional embeddings (TPE) demonstrates better performance on the structure and compositional datasets, showing that table information can be injected by with other factors such as masking and bias.

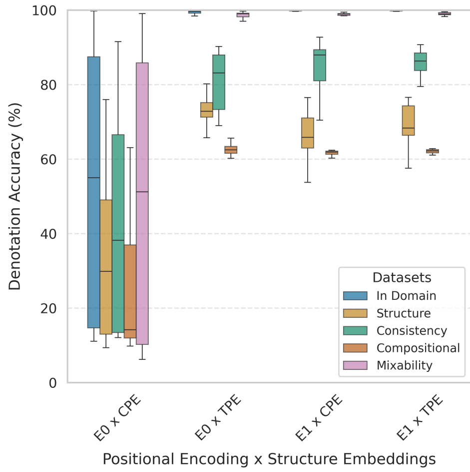

Figure 7: Impact of Positional Encoding (CPE, TPE) in Combination with Structure Embeddings (RCE, NRCE).

D Sparse Mask Details

Table 8: Performance (Structured generalization) with and without Bias for different sparsity levels.

	B0		B1	
Sparsity	**Mean**	**Var**	**Mean**	**Var**
M0	59.1	0.019	69.6	0.004
M1	75.5	0.001	75.7	0.001
M2	71.4	0.002	70.6	0.003
M3	67.5	0.000	66.9	0.001
M4*	71.4	0.008	72.8	0.002
M5*	75.6	0.000	74.2	0.000
M6*	71.5	0.000	72.0	0.001

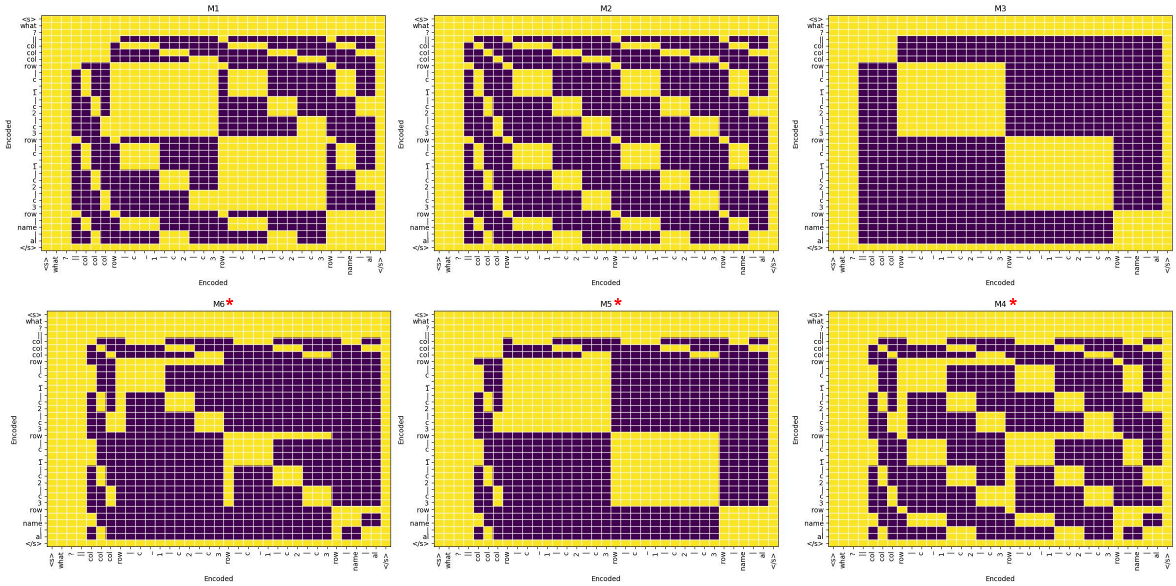

Figure 8: Visualization of sparse masks ranging from M0 (no sparsity) to M6* (high sparsity) for T2 structural token. As the mask sparsity level increases from M0 to M6, the sparsity of the mask increase. Masks marked with a red star *, such as M6*, indicate that they are only applicable to special tokens corresponding to Row-Column Cells (T2).

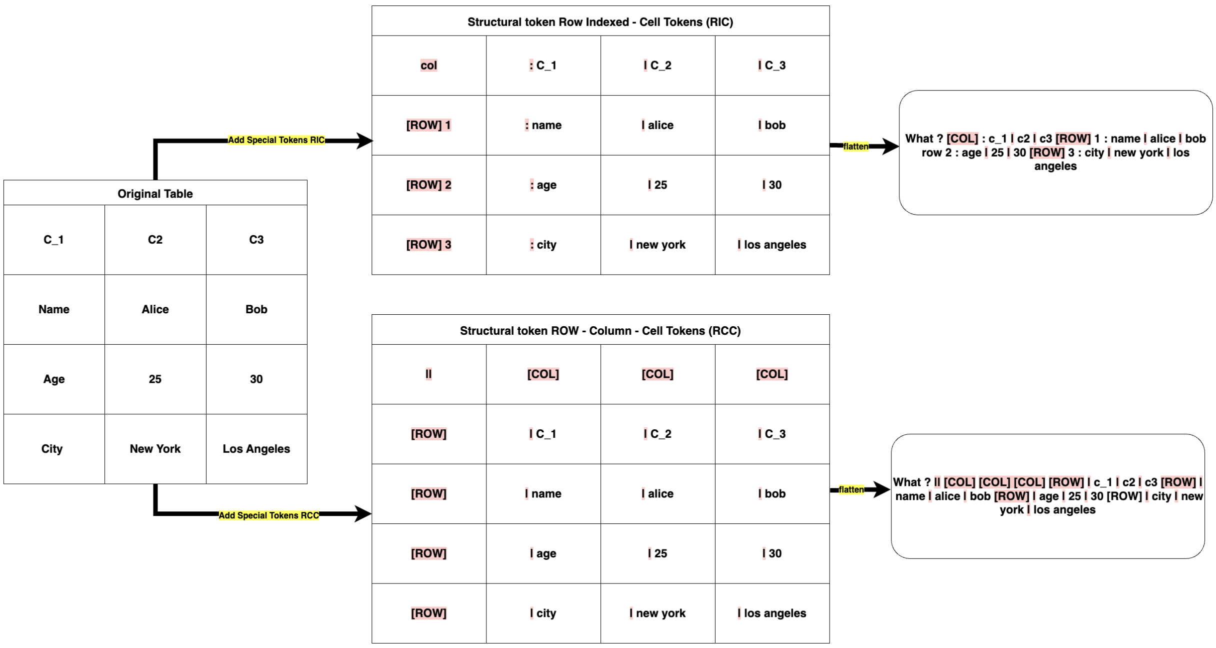

Figure 9: This figure illustrates two strategies for incorporating special tokens: T2 (Row-Column-Cell Tokens) and T1 (Row Indexed-Cell Tokens). In both approaches, special tokens are first added to the table, and then the table is flattened by concatenating each row sequentially.