TABLEFORMER: Robust Transformer Modeling for Table-Text Encoding

Jingfeng Yang* **Aditya Gupta†** **Shyam Upadhyay†**
Luheng He† **Rahul Goel†** **Shachi Paul †**
*Georgia Institute of Technology
†Google Assistant
jingfengyangpku@gmail.com
tableformer@google.com

Abstract

Understanding tables is an important aspect of natural language understanding. Existing models for table understanding require linearization of the table structure, where row or column order is encoded as an *unwanted* bias. Such spurious biases make the model vulnerable to row and column order perturbations. Additionally, prior work has not thoroughly modeled the table structures or table-text alignments, hindering the table-text understanding ability. In this work, we propose a robust and structurally aware table-text encoding architecture TABLEFORMER, where tabular structural biases are incorporated completely through learnable attention biases. TABLEFORMER is (1) strictly invariant to row and column orders, and, (2) could understand tables better due to its tabular inductive biases. Our evaluations showed that TABLEFORMER outperforms strong baselines in all settings on SQA, WTQ and TABFACT table reasoning datasets, and achieves state-of-the-art performance on SQA, especially when facing answer-invariant row and column order perturbations (6% improvement over the best baseline), because previous SOTA models' performance drops by 4% - 6% when facing such perturbations while TABLEFORMER is not affected.[1]

1 Introduction

Recently, semi-structured data (e.g. variable length tables without a fixed data schema) has attracted more attention because of its ubiquitous presence on the web. On a wide range of various table reasoning tasks, Transformer based architecture along with pretraining has shown to perform well (Eisenschlos et al., 2021; Liu et al., 2021).

In a nutshell, prior work used the Transformer architecture in a BERT like fashion by serializing tables or rows into word sequences (Yu et al., 2020; Liu et al., 2021), where original position ids are used as positional information. Due to the usage of row/column ids and global position ids, prior strategies to linearize table structures introduced spurious row and column order biases (Herzig et al., 2020; Eisenschlos et al., 2020, 2021; Zhang et al., 2020; Yin et al., 2020). Therefore, those models are vulnerable to row or column order perturbations. But, ideally, the model should make consistent predictions regardless of the row or column ordering for all practical purposes. For instance, in Figure 1, the predicted answer of TAPAS model (Herzig et al., 2020) for Question (a) *"Of all song lengths, which one is the longest?"* based on the original table is *"5:00"*, which is incorrect. However, if the first row is adjusted to the end of the table during inference,

Title	Length
Screwed Up	**5:02**
Ghetto Queen	5:00

Question: Of all song lengths, which one is the longest?
Gold Answer: 5:02
TAPAS: 5:00
TAPAS after row order perturbation: 5:02
TABLEFORMER: 5:02

(a) TAPAS predicts incorrect answer based on the original table, while it gives the correct answer if the first row is moved to the end of the table.

Nation	Gold	Silver	Bronze
Great Britain	2	1	2
Spain	**1**	**2**	**0**
Ukraine	**0**	**2**	**0**

Question: Which nation received 2 silver medals?
Gold Answer: Spain, Ukraine
TAPAS: Spain
TABLEFORMER: Spain, Ukraine
TABLEFORMER w/o a proposed structural bias: Spain

(b) TAPAS gives incomplete answer due to its limited cell grounding ability.

Figure 1: Examples showing the limitations of existing models (a) vulnerable to perturbations, and (b) lacking structural biases. In contrast, our proposed TABLEFORMER predicts correct answers for both questions.

*Work done during an internship at Google.

[1]Code has been released at https://github.com/google-research/tapas/blob/master/TABLEFORMER.md

the model gives the correct length *"5:02"* as answer. This probing example shows that the model being aware of row order information is inclined to select length values to the end of the table due to spurious training data bias. In our experiments on the SQA dataset, TAPAS models exhibit a 4% - 6% (Section 5.2) absolute performance drop when facing such answer-invariant perturbations.

Besides, most prior work (Chen et al., 2020; Yin et al., 2020) did not incorporate enough structural biases to models to address the limitation of sequential Transformer architecture, while others inductive biases which are either too strict (Zhang et al., 2020; Eisenschlos et al., 2021) or computationally expensive (Yin et al., 2020).

To this end, we propose TABLEFORMER, a Transformer architecture that is robust to row and column order perturbations, by incorporating structural biases more naturally. TABLEFORMER relies on 13 types of task-independent table↔text attention biases that respect the table structure and table-text relations. For Question (a) in Figure 1, TABLEFORMER could predict the correct answer regardless of perturbation, because the model could identify the same row information with our *"same row"* bias, avoiding spurious biases introduced by row and global positional embeddings. For Question (b), TAPAS predicted only partially correct answer, while TABLEFORMER could correctly predict *"Spain, Ukraine"* as answers. That's because our *"cell to sentence"* bias could help table cells ground to the paired sentence. Detailed attention bias types are discussed in Section 5.2.

Experiments on 3 table reasoning datasets show that TABLEFORMER consistently outperforms original TAPAS in all pretraining and intermediate pretraining settings with fewer parameters. Also, TABLEFORMER's invariance to row and column perturbations, leads to even larger improvement over those strong baselines when tested on perturbations. Our contributions are as follows:

- We identified the limitation of current table-text encoding models when facing row or column perturbation.
- We propose TABLEFORMER, which is guaranteed to be invariant to row and column order perturbations, unlike current models.
- TABLEFORMER encodes table-text structures better, leading to SoTA performance on SQA dataset, and ablation studies show the effectiveness of the introduced inductive biases.

2 Preliminaries: TAPAS for Table Encoding

In this section, we discuss TAPAS which serves as the backbone of the recent state-of-the-art table-text encoding architectures. TAPAS (Herzig et al., 2020) uses Transformer architecture in a BERT like fashion to pretrain and finetune on tabular data for table-text understanding tasks. This is achieved by using linearized table and texts for masked language model pre-training. In the fine-tuning stage, texts in the linearized table and text pairs are queries or statements in table QA or table-text entailment tasks, respectively.

Specifically, TAPAS uses the tokenized and flattened text and table as input, separated by `[SEP]` token, and prefixed by `[CLS]`. Besides token, segment, and global positional embedding introduced in BERT (Devlin et al., 2019), it also uses rank embedding for better numerical understanding. Moreover, it uses column and row embedding to encode table structures.

Concretely, for any table-text linearized sequence $S = \{v_1, v_2, \cdots, v_n\}$, where n is the length of table-text sequence, the input to TAPAS is summation of embedding of the following:

$$\begin{aligned}
\text{token ids } (W) &= \{w_{v_1}, w_{v_2}, \cdots, w_{v_n}\} \\
\text{positional ids } (B) &= \{b_1, b_2, \cdots, b_n\} \\
\text{segment ids } (G) &= \{g_{seg_1}, g_{seg_2}, \cdots, g_{seg_n}\} \\
\text{column ids } (C) &= \{c_{col_1}, c_{col_2}, \cdots, c_{col_n}\} \\
\text{row ids } (R) &= \{r_{row_1}, r_{row_2}, \cdots, r_{row_n}\} \\
\text{rank ids } (Z) &= \{z_{rank_1}, z_{rank_2}, \cdots, z_{rank_n}\}
\end{aligned}$$

where seg_i, col_i, row_i, $rank_i$ correspond to the segment, column, row, and rank id for the ith token, respectively.

As for the model, TAPAS uses BERT's self-attention architecture (Vaswani et al., 2017) off-the-shelf. Each Transformer layer includes a multi-head self-attention sub-layer, where each token attends to all the tokens. Let the layer input $H = [h_1, h_2, \cdots, h_n]^\top \in \mathbb{R}^{n \times d}$ corresponding to S, where d is the hidden dimension, and $h_i \in \mathbb{R}^{d \times 1}$ is the hidden representation at position i. For a single-head self-attention sub-layer, the input H is projected by three matrices $W^Q \in \mathbb{R}^{d \times d_K}$, $W^K \in \mathbb{R}^{d \times d_K}$, and $W^V \in \mathbb{R}^{d \times d_V}$ to the corre-

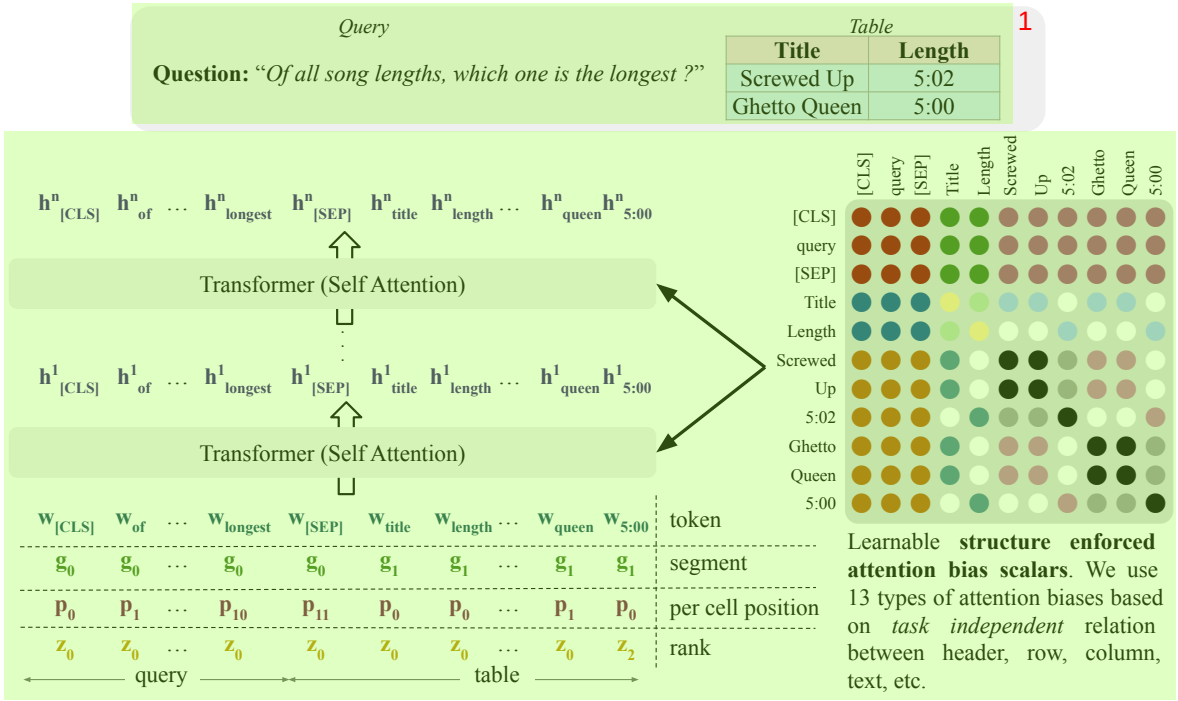

Figure 2: TABLEFORMER input and attention biases in the self attention module. This example corresponds to table (a) in Figure 1 and its paired question "query". Different colors in the attention bias matrix denote different types of task independent biases derived based on the table structure and the associated text.

sponding representations Q, K, and V:

$$Q = HW^Q, \quad V = HW^V, \quad K = HW^K \tag{1}$$

Then, the output of this single-head self-attention sub-layer is calculated as:

$$\text{Attn}(H) = \text{softmax}(\frac{QK^\top}{\sqrt{d_K}})V \tag{2}$$

3 TABLEFORMER: Robust Structural Table Encoding

As shown in Figure 2, TABLEFORMER encodes the general table structure along with the associated text by introducing task-independent relative attention biases for table-text encoding to facilitate the following: (a) structural inductive bias for better table understanding and table-text alignment, (b) robustness to table row/column perturbation.

Input of TABLEFORMER. TABLEFORMER uses the same token embeddings W, segment embeddings G, and rank embeddings Z as TAPAS. However, we make 2 major modifications:

1) No row or column ids. We do not use row embeddings R or column embeddings C to avoid any potential spurious row and column order biases.

2) Per cell positional ids. To further remove any inter-cell order information, global positional embeddings B are replaced by per cell positional embeddings $P = \{p_{pos_1}, p_{pos_2}, \cdots, p_{pos_n}\}$, where we follow Eisenschlos et al. (2021) to reset the index of positional embeddings at the beginning of each cell, and pos_i correspond to the per cell positional id for the ith token.

Positional Encoding in TABLEFORMER. Note that the Transformer model either needs to specify different positions in the input (i.e. absolute positional encoding of Vaswani et al. (2017)) or encode the positional dependency in the layers (i.e. relative positional encoding of Shaw et al. (2018)).

TABLEFORMER does not consume any sort of column and row order information in the input. The main intuition is that, for cells in the table, the only useful positional information is whether two cells are in the same row or column and the column header of each cell, instead of the absolute order of the row and column containing them. Thus, inspired by relative positional encoding (Shaw et al., 2018) and graph encoding (Ying et al., 2021), we capture this with a same column/row relation as one kind of relative position between two linearized tokens. Similarly, we uses 12 such table-text struc-

ture relevant relations (including same cell, cell to header and so on) and one extra type representing all other relations not explicitly defined. All of them are introduced in the form of learnable attention bias scalars.

Formally, we consider a function $\phi(v_i, v_j) : V \times V \rightarrow \mathbb{N}$, which measures the relation between v_i and v_j in the sequence $(v_i, v_j \in S)$. The function ϕ can be defined by any relations between the tokens in the table-text pair.

Attention Biases in TABLEFORMER. In our work, $\phi(v_i, v_j)$ is chosen from 13 bias types, corresponding to 13 table-text structural biases. The attention biases are applicable to any table-text pair and can be used for any downstream task:

- *"same row"* identifies the same row information without ordered row id embedding or global positional embedding, which help the model to be invariant to row perturbations,
- *"same column"*, *"header to column cell"*, and *"cell to column header"* incorporates the same column information without ordered column id embedding,
- *"cell to column header"* makes each cell aware of its column header without repeated column header as features,
- *"header to sentence"* and *"cell to sentence"* help column grounding and cell grounding of the paired text,
- *"sentence to header"*, *"sentence to cell"*, and *"sentence to sentence"* helps to understand the sentence with the table as context,
- *"header to same header"* and *"header to other header"* for better understanding of table schema, and *"same cell bias"* for cell content understanding.

Note that, each cell can still attend to other cells in the different columns or rows through *"others"* instead of masking them out strictly.

We assign each bias type a learnable scalar, which will serve as a bias term in the self-attention module. Specifically, each self-attention head in each layer have a set of learnable scalars $\{b_1, b_2, \cdots, b_{13}\}$ corresponding to all types of introduced biases. For one head in one self-attention sub-layer of TABLEFORMER, Equation 2 in the Transformer is replaced by:

$$\bar{A} = \frac{QK^\top}{\sqrt{d_K}}, \quad A = \bar{A} + \hat{A} \tag{3}$$

$$\text{Attn}(H) = \text{softmax}(A)V \tag{4}$$

where $\bar{A}$ is a matrix capturing the similarity between queries and keys, $\hat{A}$ is the Attention Bias Matrix, and $\hat{A}_{i,j} = b_{\phi(v_i,v_j)}$.

Relation between TABLEFORMER and ETC. ETC (Ainslie et al., 2020) uses vectors to represent relative position labels, although not directly applied to table-text pairs due to its large computational overhead (Eisenschlos et al., 2021). TABLEFORMER differs from ETC in the following aspects (1) ETC uses relative positional embeddings while TABLEFORMER uses attention bias scalars. In practice, we observed that using relative positional embeddings increases training time by more than 7x, (2) ETC uses global memory and local attention, while TABLEFORMER uses pairwise attention without any global memory overhead, (3) ETC uses local sparse attention with masking, limiting its ability to attend to all tokens, (4) ETC did not explore table-text attention bias types exhaustively. Another table encoding model MATE (Eisenschlos et al., 2021) is vulnerable to row and column perturbations, and shares limitation (3) and (4).

4 Experimental Setup

4.1 Datasets and Evaluation

We use the following datasets in our experiments.

Table Question Answering. For the table QA task, we conducted experiments on WikiTableQuestions (WTQ) (Pasupat and Liang, 2015) and Sequential QA (SQA) (Iyyer et al., 2017) datasets. WTQ was crowd-sourced based on complex questions on Wikipedia tables. SQA is composed of 6,066 question sequences (2.9 question per sequence on average), constructed by decomposing a subset of highly compositional WTQ questions.

Table-Text Entailment. For the table-text entailment task, we used TABFACT dataset (Chen et al., 2020), where the tables were extracted from Wikipedia and the sentences were written by crowd workers. Among total 118, 000 sentences, each one is a positive (entailed) or negative sentence.

Perturbation Evaluation Set. For SQA and TABFACT, we also created new test sets to measure models' robustness to answer-invariant row and column perturbations during inference. Specifically, row and column orders are randomly perturbed for all tables in the standard test sets.[2]

Pre-training All the models are first tuned on the *Wikipidia* text-table pretraining dataset (Herzig et al., 2020), optionally tuned on synthetic dataset at an intermediate stage ("inter") (Eisenschlos et al., 2020), and finally fine-tuned on the target dataset. To get better performance on WTQ, we follow Herzig et al. (2020) to further pretrain on SQA dataset after the intermediate pretraining stage in the "inter-sqa" setting.

Evaluation For SQA, we report the cell selection accuracy for all questions (ALL) using the official evaluation script, cell selection accuracy for all sequences (SEQ), and the denotation accuracy for all questions (ALL_d). To evaluate the models' robustness in the instance level after perturbations, we also report a lower bound of example prediction variation percentage:

$$VP = \frac{(\text{t2f} + \text{f2t})}{(\text{t2t} + \text{t2f} + \text{f2t} + \text{f2f})} \quad (5)$$

where t2t, t2f, f2t, and f2f represents how many example predictions turning from correct to correct, from correct to incorrect, from incorrect to correct and from incorrect to incorrect, respectively, after perturbation. We report denotation accuracy on WTQ and binary classification accuracy on TABFACT respectively.

4.2 Baselines

We use TAPAS_BASE and $\text{TAPAS}_\text{LARGE}$ as baselines, where Transformer architectures are exactly same as BERT_BASE and BERT_LARGE (Devlin et al., 2019), and parameters are initialized from BERT_BASE and BERT_LARGE respectively. Correspondingly, we have our $\text{TABLEFORMER}_\text{BASE}$ and $\text{TABLEFORMER}_\text{LARGE}$, where attention bias scalars are initialized to zero, and all other parameters are initialized from BERT_BASE and BERT_LARGE.

4.3 Perturbing Tables as Augmented Data

Could we alleviate the spurious ordering biases by data augmentation alone, without making any modeling changes? To answer this, we train another set of models by augmenting the training data for TAPAS through random row and column order perturbations.[3]

For each table in the training set, we randomly shuffle all rows and columns (including corresponding column headers), creating a new table with the same content but different orders of rows and columns. Multiple perturbed versions of the same table were created by repeating this process $\{1, 2, 4, 8, 16\}$ times with different random seeds. For table QA tasks, selected cell positions are also adjusted as final answers according to the perturbed table. The perturbed table-text pairs are then used to augment the data used to train the model. During training, the model takes data created by one specific random seed in one epoch in a cyclic manner.

	Before Perturb			After Perturb	
	ALL	SEQ	ALL_d	ALL	VP
Herzig et al. (2020)	67.2	40.4	–	–	–
Eisenschlos et al. (2020)	71.0	44.8	–	–	–
Eisenschlos et al. (2021)	71.7	46.1	–	–	–
Liu et al. (2021)	–	–	74.5	–	–
TAPAS_BASE	61.1	31.3	–	57.4	14.0%
$\text{TABLEFORMER}_\text{BASE}$	**66.7**	**39.7**	–	**66.7**	**0.2%**
$\text{TAPAS}_\text{LARGE}$	66.8	39.9	–	60.5	15.1%
$\text{TABLEFORMER}_\text{LARGE}$	**70.3**	**44.8**	–	**70.3**	**0.1%**
TAPAS_BASE inter	67.5	38.8	–	61.0	14.3%
$\text{TABLEFORMER}_\text{BASE}$ inter	**69.4**	**43.5**	–	**69.3**	**0.1%**
$\text{TAPAS}_\text{LARGE}$ inter	70.6	43.9	–	66.1	10.8%
$\text{TABLEFORMER}_\text{LARGE}$ inter	**72.4**	**47.5**	**75.9**	**72.3**	**0.1%**

Table 1: Results on SQA test set before and after perturbation during inference (median of 5 runs). ALL is cell selection accuracy, SEQ is cell selection accuracy for all question sequences, ALL_d is denotation accuracy for all questions (reported to compare with Liu et al. (2021)). VP is model prediction variation percentage after perturbation. Missing values are those not reported in the original paper.

5 Experiments and Results

Besides standard testing results to compare TABLEFORMER and baselines, we also answer the following questions through experiments:

- How robust are existing (near) state-of-the-art table-text encoding models to semantic preserving perturbations in the input?
- How does TABLEFORMER compare with existing table-text encoding models when tested

[2] We fixed perturbation random seeds to make our results reproducible.

[3] By perturbation, we mean shuffling row and columns instead of changing/swapping content blindly.

	Before Perturb					After Perturb			
	dev	test	$\text{test}_{\text{simple}}$	$\text{test}_{\text{complex}}$	$\text{test}_{\text{small}}$	test	$\text{test}_{\text{simple}}$	$\text{test}_{\text{complex}}$	$\text{test}_{\text{small}}$
Eisenschlos et al. (2020)	81.0	81.0	92.3	75.6	83.9	–	–	–	–
Eisenschlos et al. (2021)	–	81.4	–	–	–	–	–	–	–
TAPAS$_{\text{BASE}}$	72.8	72.3	84.8	66.2	74.4	71.2	83.4	65.2	72.5
TABLEFORMER$_{\text{BASE}}$	**75.1**	**75.0**	**88.2**	**68.5**	**77.1**	**75.0**	**88.2**	**68.5**	**77.1**
TAPAS$_{\text{LARGE}}$	74.7	74.5	86.6	68.6	76.8	73.7	86.0	67.7	76.1
TABLEFORMER$_{\text{LARGE}}$	**77.2**	**77.0**	**90.2**	**70.5**	**80.3**	**77.0**	**90.2**	**70.5**	**80.3**
TAPAS$_{\text{BASE}}$ inter	78.4	77.9	90.1	71.9	80.5	76.8	89.5	70.5	79.7
TABLEFORMER$_{\text{BASE}}$ inter	**79.7**	**79.2**	**91.6**	**73.1**	**81.7**	**79.2**	**91.6**	**73.1**	**81.7**
TAPAS$_{\text{LARGE}}$ inter	80.6	80.6	92.0	74.9	83.1	79.2	91.7	73.0	83.0
TABLEFORMER$_{\text{LARGE}}$ inter	**82.0**	**81.6**	**93.3**	**75.9**	**84.6**	**81.6**	**93.3**	**75.9**	**84.6**

Table 2: Binary classification accuracy on TABFACT development and 4 splits of test set, as well as performance on test sets with our perturbation evaluation. Median of 5 independent runs are reported. Missing values are those not reported in the original paper.

Model	dev	test
Herzig et al. (2020)	–	48.8
Eisenschlos et al. (2021)	–	51.5
TAPAS$_{\text{BASE}}$	23.6	24.1
TABLEFORMER$_{\text{BASE}}$	**34.4**	**34.8**
TAPAS$_{\text{LARGE}}$	40.8	41.7
TABLEFORMER$_{\text{LARGE}}$	**42.5**	**43.9**
TAPAS$_{\text{BASE}}$ inter-sqa	44.8	45.1
TABLEFORMER$_{\text{BASE}}$ inter-sqa	**46.7**	**46.5**
TAPAS$_{\text{LARGE}}$ inter-sqa	49.9	50.4
TABLEFORMER$_{\text{LARGE}}$ inter-sqa	**51.3**	**52.6**

Table 3: Denotation accuracy on WTQ development and test set. Median of 5 independent runs are reported.

on similar perturbations, both in terms of performance and robustness?

- Can we use perturbation based data augmentation to achieve robustness at test time?
- Which attention biases in TABLEFORMER contribute the most to performance?

5.1 Main Results

Table 1, 2, and 3 shows TABLEFORMER performance on SQA, TABFACT, and WTQ, respectively. As can be seen, TABLEFORMER outperforms corresponding TAPAS baseline models in all settings on SQA and WTQ datasets, which shows the general effectiveness of TABLEFORMER's structural biases in Table QA datasets. Specifically, TABLEFORMER$_{\text{LARGE}}$ combined with intermediate pretraining achieves new state-of-the-art performance on SQA dataset.

Similarly, Table 2 shows that TABLEFORMER also outperforms TAPAS baseline models in all settings, which shows the effectiveness of TABLEFORMER in the table entailment task. Note that, Liu et al. (2021) is not comparable to our results, because they used different pretraining data, different pretraining objectives, and BART NLG model instead of BERT NLU model. But TABLEFORMER attention bias is compatible with BART model.

5.2 Perturbation Results

One of our major contributions is to systematically evaluate models' performance when facing row and column order perturbation in the testing stage.

Ideally, model predictions should be consistent on table QA and entailment tasks when facing such perturbation, because the table semantics remains the same after perturbation.

However, in Table 1 and 2, we can see that in our perturbed test set, performance of all TAPAS models drops significantly in both tasks. TAPAS models drops by at least 3.7% and up to 6.5% in all settings on SQA dataset in terms of ALL accuracy, while our TABLEFORMER being strictly invariant to such row and column order perturbation leads to no drop in performance.[4] Thus, in the perturbation setting, TABLEFORMER outperforms all TAPAS baselines even more significantly, with at least 6.2% and 2.4% improvements on SQA and TABFACT dataset, respectively. In the instance level, we can see that, with TAPAS, there are many example predictions changed due to high VP, while there is nearly no

[4] In SQA dataset, there is at most absolute 0.1% performance drop because of some bad data point issues. Specifically, some columns in certain tables are exactly the same, but the ground-truth selected cells are in only one of such columns. TABLEFORMER would select from one column randomly.

Model	Number of parameters
TAPAS$_{\text{BASE}}$	110 M
TABLEFORMER$_{\text{BASE}}$	110 M - 2*512*768 + 12*12*13 = 110 M - 0.8 M + 0.002 M
TAPAS$_{\text{LARGE}}$	340 M
TABLEFORMER$_{\text{LARGE}}$	340 M - 2*512*1024 + 24*16*13 = 340 M - 1.0 M + 0.005M

Table 4: Model size comparison.

example predictions changed with TABLEFORMER (around zero VP).

5.3 Model Size Comparison

We compare the model sizes of TABLEFORMER and TAPAS in Table 4. We added only a few attention bias scalar parameters (13 parameters per head per layer) in TABLEFORMER, which is negligible compared with the BERT model size. Meanwhile, we delete two large embedding metrics (512 row ids and 512 column ids). Thus, TABLEFORMER outperforms TAPAS with fewer parameters.

5.4 Analysis of TABLEFORMER Submodules

In this section, we experiment with several variants of TABLEFORMER to understand the effectiveness of its submodules. The performance of all variants of TAPAS and TABLEFORMER that we tried on the SQA development set is shown in Table 5.

Learnable Attention Biases v/s Masking. Instead of adding learnable bias scalars, we mask out some attention scores to restrict attention to those tokens in the same columns and rows, as well as the paired sentence, similar to Zhang et al. (2020) (SAT). We can see that TAPAS$_{\text{BASE-SAT}}$ performs worse than TAPAS$_{\text{BASE}}$, which means that restricting attention to only same columns and rows by masking reduce the modeling capacity. This led to choosing soft bias addition over hard masking.

Attention Bias Scaling. Unlike TABLEFORMER, we also tried to add attention biases before the scaling operation in the self-attention module (SO). Specifically, we compute pair-wise attention score by:

$$A_{ij} = \frac{(h_i^\top W^Q)(h_j^\top W^K)^\top + \hat{A}_{ij}}{\sqrt{d_K}} \quad (6)$$

	rc-gp	*c-gp*	*gp*	*pcp*
TAPAS$_{\text{BASE}}$	57.6	47.4	46.4	29.1
TAPAS$_{\text{BASE-SAT}}$	45.2	-	-	-
TABLEFORMER$_{\text{BASE-SO}}$	60.0	60.2	59.8	60.7
TABLEFORMER$_{\text{BASE}}$	62.2	61.5	61.7	61.9

Table 5: ALL questions' cell selection accuracy of TABLEFORMER variants on SQA development set. *rc-gp* represents the setting including row ids, column ids and global positional ids, *c-gp* represents column ids and global positional ids, *gp* represents global positional ids, and *pcp* represents per-cell positional ids. "SAT" represents masking out some attention scores. "SO" represents adding attention bias before scaling.

instead of using:

$$A_{ij} = \frac{(h_i^\top W^Q)(h_j^\top W^K)^\top}{\sqrt{d_K}} + \hat{A}_{ij}, \quad (7)$$

which is the element-wise version of Equation 1 and 3. However, Table 5 shows that TABLEFORMER$_{\text{BASE-SO}}$ performs worse than TABLEFORMER$_{\text{BASE}}$, showing the necessity of adding attention biases after the scaling operation. We think the reason is that the attention bias term does not require scaling, because attention bias scalar magnitude is independent of d_K, while the dot products grow large in magnitude for large values of d_K. Thus, such bias term could play an more important role without scaling, which helps each attention head know clearly what to pay more attention to according to stronger inductive biases.

Row, Column, & Global Positional IDs. With TAPAS$_{\text{BASE}}$, TABLEFORMER$_{\text{BASE-SO}}$, and TABLEFORMER$_{\text{BASE}}$, we first tried the full-version where row ids, column ids, and global positional ids exist as input (*rc-gp*). Then, we deleted row ids (*c-gp*), and column ids (*gp*) sequentially. Finally, we changed global positional ids in *gp* to per-cell positional ids (*pcp*). Table 5 shows that TAPAS$_{\text{BASE}}$ performs significantly worse from *rc-gp* $\rightarrow$ *c-gp* $\rightarrow$ *gp* $\rightarrow$ *pcp*, because table structure information are deleted sequentially during such process. However, with TABLEFORMER$_{\text{BASE}}$, there is no obvious performance drop during the same process. That shows the structural inductive biases in TABLEFORMER can provide complete table structure information. Thus, row ids, column ids and global positional ids are not necessary in TABLEFORMER. We pick TABLEFORMER *pcp* setting as our final version to conduct all other experiments in this paper. In this way, TABLEFORMER is strictly

	Befor Perturb		After Perturb	
	ALL	SEQ	ALL	VP
TAPAS$_{\text{BASE}}$	61.1	31.3	57.4	14.0%
TAPAS$_{\text{BASE}}$ 1p	63.4	34.6	63.4	9.9%
TAPAS$_{\text{BASE}}$ 2p	64.6	35.6	64.5	8.4%
TAPAS$_{\text{BASE}}$ 4p	65.1	37.0	65.0	8.1%
TAPAS$_{\text{BASE}}$ 8p	65.1	37.3	64.3	7.2%
TAPAS$_{\text{BASE}}$ 16p	62.4	33.6	62.2	7.0%
TABLEFORMER$_{\text{BASE}}$	**66.7**	**39.7**	**66.7**	**0.1%**

Table 6: Comparison of TABLEFORMER and perturbed data augmentation on SQA test set, where VP represents model prediction variation percentage after perturbation. Median of 5 independent runs are reported.

	ALL	SEQ
TABLEFORMER$_{\text{BASE}}$	**62.1**	**38.4**
- Same Row	32.1	2.8
- Same Column	**62.1**	37.7
- Same Cell	61.8	**38.4**
- Cell to Column Header	60.7	36.6
- Cell to Sentence	60.5	36.4
- Header to Column Cell	60.5	35.8
- Header to Other Header	60.6	35.8
- Header to Same Header	61.0	36.9
- Header to Sentence	61.1	36.3
- Sentence to Cell	60.8	36.2
- Sentence to Header	61.0	37.3
- Sentence to Sentence	60.0	35.3
- All Column Related (# 2, 4, 6)	54.5	29.3

Table 7: Ablation study of proposed attention biases.

invariant to row and column order perturbation by avoiding spurious biases in those original ids.

5.5 Comparison of TABLEFORMER and Perturbed Data Augmentation

As stated in Section 4.3, perturbing row and column orders as augmented data during training can serve as another possible solution to alleviate the spurious row/column ids bias. Table 6 shows the performance of TABPAS$_{\text{BASE}}$ model trained with additional {1, 2, 4, 8, 16} perturbed versions of each table as augmented data.

We can see that the performance of TAPAS$_{\text{BASE}}$ on SQA dataset improves with such augmentation. Also, as the number of perturbed versions of each table increases, model performance first increases and then decreases, reaching the best results with 8 perturbed versions. We suspect that too many versions of the same table confuse the model about different row and column ids for the same table, leading to decreased performance from 8p to 16p. Despite its usefulness, such data perturbation is still worse than TABLEFORMER, because it could not incorporate other relevant text-table structural inductive biases like TABLEFORMER.

Although, such data augmentation makes the model more robust to row and column order perturbation with smaller VP compared to standard TAPAS$_{\text{BASE}}$, there is still a significant prediction drift after perturbation. As shown in Table 6, VP decreases from 1p to 16p, however, the best VP (7.0%) is still much higher than (nearly) no variation (0.1%) of TABLEFORMER.

To sum up, TABLEFORMER is superior to row and column order perturbation augmentation, because of its additional structural biases and strictly consistent predictions after perturbation.

5.6 Attention Bias Ablation Study

We conduct ablation study to demonstrate the utility of all 12 types of defined attention biases. For each ablation, we set the corresponding attention bias type id to *"others"* bias id. Table 7 shows TAPAS$_{\text{BASE}}$'s performance SQA dev set. Overall, all types of attention biases help the TABLEFORMER performance to some extent, due to certain performance drop after deleting each bias type.

Amongst all the attention biases, deleting *"same row"* bias leads to most significant performance drop, showing its crucial role for encoding table row structures. There is little performance drop after deleting *"same column"* bias, that's because TABLEFORMER could still infer the same column information through *"cell to its column header"* and *"header to its column cell"* biases. After deleting all same column information (*"same column"*, *"cell to column header"* and *"header to column cell"* biases), TABLEFORMER performs significantly worse without encoding column structures. Similarly, there is little performance drop after deleting *"same cell"* bias, because TABLEFORMER can still infer same cell information through *"same row"* and *"same column"* biases.

5.7 Limitations of TABLEFORMER

TABLEFORMER increases the training time by around 20%, which might not be ideal for very long tables and would require a scoped approach. Secondly, with the strict row and column order invariant property, TABLEFORMER cannot deal with questions based on absolute orders of rows in tables. This however is not a practical requirement based on the current dataset. Doing a manual study of 1800 questions in SQA dataset, we found that

there are 4 questions[5] (0.2% percentage) whose answers depend on orders of rows. Three of them asked *"which one is at the top of the table"*, another asks *"which one is listed first"*. However, these questions could be potentially answered by adding back row and column order information based on TABLEFORMER.

6 Other Related Work

Transformers for Tabular Data. Yin et al. (2020) prepended corresponding column headers to cells contents, and Chen et al. (2020) used corresponding column headers as features for cells. However, such methods encode each table header multiple times, leading to duplicated computing overhead. Also, tabular structures (e.g. same row information) are not fully incorporated to such models. Meanwhile, Yin et al. (2020) leveraged row encoder and column encoder sequentially, which introduced much computational overhead, thus requiring retrieving some rows as a preprocessing step. Finally, SAT (Zhang et al., 2020), Deng et al. (2021) and Wang et al. (2021) restricted attention to same row or columns with attention mask, where such inductive bias is too strict that cells could not directly attend to those cells in different row and columns, hindering the modeling ability according to Table 5. Liu et al. (2021) used the seq2seq BART generation model with a standard Transformer encoder-decoder architecture. In all models mentioned above, spurious inter-cell order biases still exist due to global positional ids of Transformer, leading to the vulnerability to row or column order perturbations, while our TABLEFORMER could avoid such problem. Mueller et al. (2019) and Wang et al. (2020) also used relative positional encoding to encode table structures, but they modeled the relations as learnable relation vectors, whose large overhead prevented pretraining and led to poor performance without pretraining, similarly to ETC (Ainslie et al., 2020) explained in Section 3.

Structural and Relative Attention. Modified attention scores has been used to model relative positions (Shaw et al., 2018), long documents (Dai et al., 2019; Beltagy et al., 2020; Ainslie et al., 2020), and graphs (Ying et al., 2021). But adding learnable attention biases to model tabular structures has been under-explored.

[5]We find such 4 questions by manually looking at all 125 questions where the model predictions turn from correct to incorrect after replacing $\text{TAPAS}_{\text{LARGE}}$ with $\text{TABLEFORMER}_{\text{LARGE}}$.

7 Conclusion

In this paper, we identified the vulnerability of prior table encoding models along two axes: (a) capturing the structural bias, and (b) robustness to row and column perturbations. To tackle this, we propose TABLEFORMER, where learnable task-independent learnable structural attention biases are introduced, while making it invariant to row/column order at the same time. Experimental results showed that TABLEFORMER outperforms strong baselines in 3 table reasoning tasks, achieving state-of-the-art performance on SQA dataset, especially when facing row and column order perturbations, because of its invariance to row and column orders.

Acknowledgments

We thank Julian Eisenschlos, Ankur Parikh, and the anonymous reviewers for their feedbacks in improving this paper.

Ethical Considerations

The authors foresee no ethical concerns with the research presented in this paper.

References

Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. 2020. ETC: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 268–284, Online. Association for Computational Linguistics.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv:2004.05150*.

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020. Tabfact : A large-scale dataset for table-based fact verification. In *International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy. Association for Computational Linguistics.

Xiang Deng, Huan Sun, Alyssa Lees, You Wu, and Cong Yu. 2021. TURL: Table Understanding through Representation Learning. In *VLDB*. 1

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics. 2

Julian Eisenschlos, Maharshi Gor, Thomas Müller, and William Cohen. 2021. MATE: Multi-view attention for table transformer efficiency. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7606–7619, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. 3

Julian Eisenschlos, Syrine Krichene, and Thomas Müller. 2020. Understanding tables with intermediate pre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 281–296, Online. Association for Computational Linguistics. 4

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics. 5

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1821–1831, Vancouver, Canada. Association for Computational Linguistics. 6

Qian Liu, Bei Chen, Jiaqi Guo, Zeqi Lin, and Jianguang Lou. 2021. Tapex: Table pre-training via learning a neural sql executor. *arXiv preprint arXiv:2107.07653*. 7

Thomas Mueller, Francesco Piccinno, Peter Shaw, Massimo Nicosia, and Yasemin Altun. 2019. Answering conversational questions on structured data without logical forms. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5902–5910, Hong Kong, China. Association for Computational Linguistics. 8

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics. 9 10

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics. 11

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008. 12

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online. Association for Computational Linguistics. 13

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. 14

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426, Online. Association for Computational Linguistics. 15

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do Transformers Really Perform Bad for Graph Representation? *arXiv preprint arXiv:2106.05234*. 16

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. 2020. GraPPa: Grammar-Augmented Pre-Training for Table Semantic Parsing. *arXiv preprint arXiv:2009.13845*. 17

Hongzhi Zhang, Yingyao Wang, Sirui Wang, Xuezhi Cao, Fuzheng Zhang, and Zhongyuan Wang. 2020. Table fact verification with structure-aware transformer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1624–1629, Online. Association for Computational Linguistics. 18