TABLEFORMER: Robust Transformer Modeling for Table-Text Encoding

Anonymous ACL submission

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 explicitly modeled the table structure, hindering the table-text modeling 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 strictly invariant to row and column orders, and 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 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 (Eisenirschlos 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; Eisenirschlos 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, 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...

Table: Song Lengths

<table>
<thead>
<tr>
<th>Title</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Screwed Up</td>
<td>5:02</td>
</tr>
<tr>
<td>Ghetto Queen &amp; NOVA</td>
<td>5:00</td>
</tr>
</tbody>
</table>

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.

Table: Medal Counts

<table>
<thead>
<tr>
<th>Nation</th>
<th>Gold</th>
<th>Silver</th>
<th>Bronze</th>
</tr>
</thead>
<tbody>
<tr>
<td>Great Britain</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Spain</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Ukraine</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

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.
to spurious training data bias. In our experiments on the SQA dataset, TAPAS models exhibit a 4% - 6% absolute performance drop when facing such answer-invariant perturbations.

Besides, most prior work (Chen et al., 2019; 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.

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 most inductive biases.

2 Preliminaries: TAPAS for Table Encoding

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., 2018), it also uses rank embedding for better numerical understanding. Moreover, it uses column and row id embedding to encode table structures.

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

- **token id**: \( W = \{w_{v_1}, w_{v_2}, \ldots, w_{v_n}\} \),
- **global positional id**: \( B = \{b_1, b_2, \ldots, b_n\} \),
- **segment id**: \( G = \{g_{seg_1}, g_{seg_2}, \ldots, g_{seg_n}\} \),
- **column id**: \( C = \{c_{col_1}, c_{col_2}, \ldots, c_{col_n}\} \),
- **row id**: \( R = \{r_{row_1}, r_{row_2}, \ldots, r_{row_n}\} \), and
- **rank id**: \( Z = \{z_{rank_1}, z_{rank_2}, \ldots, z_{rank_n}\} \),

where \( \text{seg}_i, c_{col_i}, r_{row_i}, \text{rank}_i \) correspond to the segment, column, row, and rank id for the \( i \)th token, respectively.

As for the model, TAPAS uses BERT 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 \( H = [h_1, h_2, \ldots, h_n]^T \in \mathbb{R}^{n \times d} \) 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 corresponding 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:

\[
\text{Attn}(H) = \text{softmax} \left( \frac{QK^\top}{\sqrt{d_K}} \right)V \tag{2}
\]

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 $P = \{p_{pos1}, p_{pos2}, \ldots, p_{pos_n}\}$, where we follow Eisenschlos et al. (2021) to reset the index of positional embeddings at the beginning of each cell, and $p_{posi}$ correspond to the per cell positional id for the $i$th 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 structure 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 $\phi(v_i, v_j) : V \times V \to \mathbb{N}$, which measures the relation between $v_i$ and $v_j$ in the sequence $(v_i, v_j \in S)$. The function $\phi$ 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 type ids, corresponding to 13 table-text structural biases. All the chosen 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 id 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 \( \{b_1, b_2, \ldots, b_{13}\} \) 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:

\[
\tilde{A} = \frac{QK^\top}{\sqrt{d_K}}, \quad A = \tilde{A} + \hat{A}
\]

\[\text{Attn}(H) = \text{softmax}(A)V\]

where \( \tilde{A} \) is a matrix capturing the similarity between queries and keys, \( \hat{A} \) is the Attention Bias Matrix, and \( \hat{A}_{i,j} = b_\phi(i_1, a_1) \).

**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 observe a large overhead by using relative positional embeddings (more than 7× training time), (2) ETC uses global memory and local attention, while TABLEFORMER uses pairwise attention without any global memory overhead, (3) ETC leverages 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) (4).

### 4 Experimental Setup

**4.1 Datasets and Evaluation**

**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., 2019), 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.\(^1\)

**Pre-training** All the models are first tuned on the Wikipedia text-table pretraining dataset, 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 (ALLd). To evaluate the models’ robustness in the instance level after perturbations, we also report a lower bound of example prediction variation percentage: \( \nu_p = (t_{2f} + f_{2f})/(t_{2t} + t_{2f} + f_{2t} + f_{2f}) \), where \( t_{2t}, t_{2f}, f_{2t}, \) and \( f_{2f} \) 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. On WTQ, we report denotation accuracy. Binary classification accuracy is reported on TABFACT.

---

\(^1\)We fixed perturbation random seeds as 4 to make our results reproducible.
4.2 Baselines

We use TAPAS\textsubscript{BASE} and TAPAS\textsubscript{LARGE} as baselines, where Transformer architectures are exactly same as BERT\textsubscript{BASE} and BERT\textsubscript{LARGE} (Devlin et al., 2018), and parameters are initialized from BERT\textsubscript{BASE} and BERT\textsubscript{LARGE} respectively. Correspondingly, we have our TABLEFORMER\textsubscript{BASE} and TABLEFORMER\textsubscript{LARGE}, where attention bias scalars are initialized to zero, and all other parameters are initialized from BERT\textsubscript{BASE} and BERT\textsubscript{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.

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.

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 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?

\(2\) By perturbation, we mean shuffling row and columns instead of changing/swapping content blindly.

<table>
<thead>
<tr>
<th>Before Perturb</th>
<th>After Perturb</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ALL</strong></td>
<td><strong>SEQ</strong></td>
</tr>
<tr>
<td>Herzig et al. (2020)</td>
<td>67.2</td>
</tr>
<tr>
<td>Eisenschlos et al. (2020)</td>
<td>71.0</td>
</tr>
<tr>
<td>Eisenschlos et al. (2021)</td>
<td>71.7</td>
</tr>
<tr>
<td>Liu et al. (2021)</td>
<td>–</td>
</tr>
<tr>
<td>TAPAS\textsubscript{BASE}</td>
<td>61.1</td>
</tr>
<tr>
<td>TABLEFORMER\textsubscript{BASE}</td>
<td>66.7</td>
</tr>
<tr>
<td>TAPAS\textsubscript{LARGE}</td>
<td>66.8</td>
</tr>
<tr>
<td>TABLEFORMER\textsubscript{LARGE}</td>
<td>70.3</td>
</tr>
<tr>
<td>TAPAS\textsubscript{LARGE} inter</td>
<td>67.5</td>
</tr>
<tr>
<td>TABLEFORMER\textsubscript{LARGE} inter</td>
<td>69.4</td>
</tr>
<tr>
<td>TAPAS\textsubscript{LARGE} inter</td>
<td>70.6</td>
</tr>
<tr>
<td>TABLEFORMER\textsubscript{LARGE} inter</td>
<td>72.4</td>
</tr>
</tbody>
</table>

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

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\textsubscript{LARGE} combined with intermediate pretraining achieves new state-of-the-art performance on SQA dataset.

Similarly, in Table 2, we can see 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. Thus, we would test TABLEFORMER with Liu et al. (2021)’s SOTA method on TABFACT and WTQ dataset in the future.

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...
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.

<table>
<thead>
<tr>
<th>Model</th>
<th>dev</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eisenschlos et al. (2020)</td>
<td>81.0</td>
<td>81.0</td>
</tr>
<tr>
<td>Eisenschlos et al. (2021)</td>
<td>81.4</td>
<td>92.3</td>
</tr>
<tr>
<td>TAPASBASE</td>
<td>72.8</td>
<td>75.1</td>
</tr>
<tr>
<td>TABLEFORMERBASE</td>
<td>75.1</td>
<td>75.0</td>
</tr>
<tr>
<td>TAPASLARGE</td>
<td>74.7</td>
<td>77.2</td>
</tr>
<tr>
<td>TABLEFORMERLARGE</td>
<td>77.2</td>
<td>77.0</td>
</tr>
<tr>
<td>TAPASBASE inter</td>
<td>78.4</td>
<td>79.7</td>
</tr>
<tr>
<td>TABLEFORMERBASE inter</td>
<td>79.7</td>
<td>79.2</td>
</tr>
<tr>
<td>TAPASLARGE inter</td>
<td>80.6</td>
<td>82.0</td>
</tr>
<tr>
<td>TABLEFORMERLARGE inter</td>
<td>82.0</td>
<td>81.6</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Model</th>
<th>dev</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Herzig et al. (2020)</td>
<td>–</td>
<td>48.8</td>
</tr>
<tr>
<td>Eisenschlos et al. (2021)</td>
<td>–</td>
<td>51.5</td>
</tr>
<tr>
<td>TAPASBASE</td>
<td>23.6</td>
<td>24.1</td>
</tr>
<tr>
<td>TABLEFORMERBASE</td>
<td>34.4</td>
<td>34.8</td>
</tr>
<tr>
<td>TAPASLARGE</td>
<td>40.8</td>
<td>41.7</td>
</tr>
<tr>
<td>TABLEFORMERLARGE</td>
<td>42.5</td>
<td>43.9</td>
</tr>
<tr>
<td>TAPASBASE inter-sqa</td>
<td>44.8</td>
<td>45.1</td>
</tr>
<tr>
<td>TABLEFORMERBASE inter-sqa</td>
<td>46.7</td>
<td>46.5</td>
</tr>
<tr>
<td>TAPASLARGE inter-sqa</td>
<td>49.9</td>
<td>50.4</td>
</tr>
<tr>
<td>TABLEFORMERLARGE inter-sqa</td>
<td>51.3</td>
<td>52.6</td>
</tr>
</tbody>
</table>

Table 4: Model size comparison.

<table>
<thead>
<tr>
<th>Model</th>
<th>Number of parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAPASBASE</td>
<td>110 M</td>
</tr>
<tr>
<td>TABLEFORMERBASE</td>
<td>110 M - 2<em>512</em>768 = 110 M - 0.8 M + 0.002 M</td>
</tr>
<tr>
<td>TAPASLARGE</td>
<td>340 M</td>
</tr>
<tr>
<td>TABLEFORMERLARGE</td>
<td>340 M - 2<em>512</em>1024 = 340 M - 1.0 M + 0.005M</td>
</tr>
</tbody>
</table>

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 TAPASBASE-SAT performs

---

1In SQA dataset, there is at most absolute 0.1% performance drop because of some bad data point issues.
worse than TAPAS_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 TABLE-FORMER, 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:

\[
A_{ij} = \frac{(h_i^Tw^Q)(h_j^Tw^K)}{\sqrt{d_K}} + \hat{A}_{ij}
\]

instead of using:

\[
A_{ij} = \frac{(h_i^Tw^Q)(h_j^Tw^K)}{\sqrt{d_K}} + \hat{A}_{ij},
\]

which is the element-wise version of Equation 1 and 3. However, Table 5 shows that TABLE-FORMER_BASE-SO performs worse than TABLE-FORMER_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.

**Effect of Row, Column, and Global Positional IDs.** With TAPAS_BASE, TABLE-FORMER_BASE-SO, and TABLE-FORMER_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”). In Table 5, we can see that TAPAS_BASE performs significantly worse from rc-gp to c-gp, to gp, to pcp, because attention bias scaling reduce the modeling capacity. This led to TABLE-FORMER_BASE-SO representations model prediction variation percentage after perturbation. Median of 5 independent runs are reported.

### Table 5: ALL questions’ cell selection accuracy of TABLE-FORMER variants on SQA development set.

<table>
<thead>
<tr>
<th></th>
<th>re-gp</th>
<th>c-gp</th>
<th>gp</th>
<th>pcp</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAPAS_BASE</td>
<td>57.6</td>
<td>47.4</td>
<td>46.4</td>
<td>29.1</td>
</tr>
<tr>
<td>TAPAS_BASE-SAT</td>
<td>45.2</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TABLEFORMER_BASE-SO</td>
<td>60.0</td>
<td>60.2</td>
<td>59.8</td>
<td>60.7</td>
</tr>
<tr>
<td>TABLEFORMER_BASE</td>
<td>62.2</td>
<td>61.5</td>
<td>61.7</td>
<td>61.9</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th>Before Perturb</th>
<th>After Perturb</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ALL</td>
<td>SEQ</td>
</tr>
<tr>
<td>TAPAS_BASE</td>
<td>61.1</td>
<td>31.3</td>
</tr>
<tr>
<td>TAPAS_BASE 1p</td>
<td>63.4</td>
<td>34.6</td>
</tr>
<tr>
<td>TAPAS_BASE 2p</td>
<td>64.6</td>
<td>35.6</td>
</tr>
<tr>
<td>TAPAS_BASE 4p</td>
<td>65.1</td>
<td>37.0</td>
</tr>
<tr>
<td>TAPAS_BASE 8p</td>
<td>65.1</td>
<td>37.3</td>
</tr>
<tr>
<td>TAPAS_BASE 16p</td>
<td>62.4</td>
<td>33.6</td>
</tr>
</tbody>
</table>

TABLEFORMER_BASE | 66.7 | 39.7 | 66.7 | 0.1% |

5.5 Comparison of TABLE-FORMER 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 TAPAS_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_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 TABLE-FORMER, because it could...
(7.0%) is still much higher than (nearly) no varia-

tions, and Table 5 shows how much our 

our T

TABLEFORMER is superior to row and 

and column order perturbation augmentation, be-

cause of its additional structural biases and strictly 

consistent predictions after perturbation.

5.6 Attention Bias Ablation Study

We conduct ablation study to demonstrate the util-

ity 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 

TAPASBASE’s performance SQA dev set. Over-

all, all types of attention biases help the TABLE-

FORMER 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 col-


table | ALL | SEQ

| TABLEFORMERBASE | 62.1 | 38.4 |

- Same Row | 32.1 | 2.8 |
- Same Column | 62.1 | 37.7 |
- Same Cell | 61.8 | 36.4 |
- 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.

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 per-

turbation with smaller \( vp \) compared to standard 

TAPASBASE; 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 vari-

ation (0.1%) of TABLEFORMER.

To sum up, TABLEFORMER is superior to row 

and column order perturbation augmentation, be-

cause of its additional structural biases and strictly 

consistent predictions after perturbation.

6 Other Related Work

Transformers for Tabular Data. Yin et al. 

(2020) prepended corresponding column headers 

to cells contents, and Chen et al. (2019) used cor-

responding 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 mod-

els. Meanwhile, Yin et al. (2020) leveraged row 

encoder and column encoder sequentially, which 

introduced much computational overhead, thus re-

quiring retrieving some rows as a preprocessing 

step. Finally, SAT (Zhang et al., 2020) 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 dif-

ferent row and columns, hindering the modeling 

ability according to Table 5. In all of those models 

mentioned above including Liu et al. (2021), spuri-

ous inter-cell order biases still exist due to global 

positional ids of Transformer, leading to the vulner-

ability to row or column order perturbations, while 

our TABLEFORMER could avoid such problem.

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., 

2019), and graphs (Ying et al., 2021). But adding 

learnable attention biases to model tabular struc-

tures has been under-explored.

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 bi-

ases 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 per-

turbations, because of its invariance to row and 

column orders.
A 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 could not 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\(^4\) (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 ids based on TABLEFORMER.

\(^4\)We find such 4 questions by manually looking at all 125 questions where the model predictions turn from correct to incorrect after replacing TAPEX with TABLEFORMER\(_{\text{LARGE}}\).