Robust (Controlled) Table-to-Text Generation with Structure-Aware Equivariance Learning

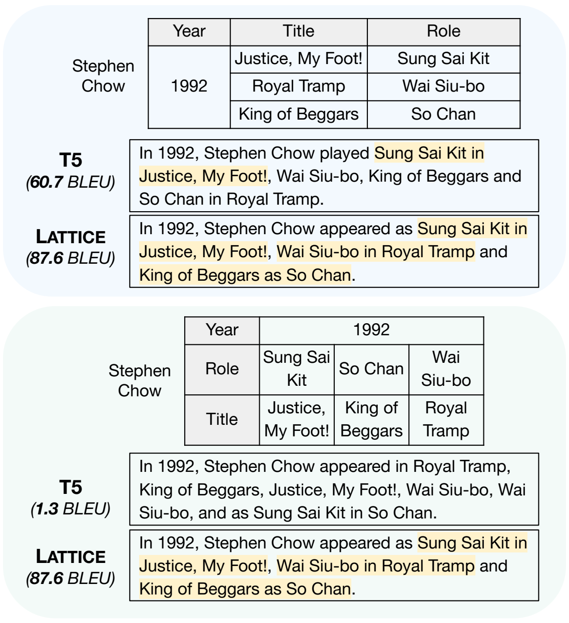
Anonymous ACL submission

Abstract

Controlled table-to-text generation seeks to generate natural language descriptions for highlighted subparts of a table. Previous SOTA systems still employ a sequence-to-sequence generation method, which merely captures the table as a linear structure and is brittle when table layouts change. We seek to go beyond this paradigm by (1) effectively expressing the relations of content pieces in the table, and (2) making our model robust to content-invariant structural transformations. Accordingly, we propose an equivariance learning framework, encoding tables with a structure-aware self-attention mechanism. This prunes the full self-attention structure into an order-invariant graph attention that captures the connected graph structure of cells belonging to the same row or column, and it differentiates between relevant cells and irrelevant cells from the structural perspective. Our framework also modifies the positional encoding mechanism to preserve the relative position of tokens in the same cell but enforce position invariance among different cells. Our technology is free to be plugged into existing table-to-text generation models, and has improved T5-based models to offer better performance on ToTTo and HiTab. Moreover, on a harder version of ToTTo, we preserve promising performance, while previous SOTA systems, even with transformation-based data augmentation, have seen significant performance drops.[1]

1 Introduction

Table-to-text generation seeks to generate natural language descriptions for content and entailed conclusions in tables. It is an important task that not only makes ubiquitous tabular data more discoverable and accessible, but also supports downstream tasks of tabular semantic retrieval (Wang et al., 2021a), reasoning (Gupta et al., 2020), fact checking (Chen et al., 2019; Wang et al., 2021b) and table-assisted question answering (Chen et al., 2020c). While rich and diverse facts can be presented in a table, the controlled table-to-text generation task, which generates focused textual descriptions for highlighted subparts of a table, has garnered much attention recently (Parikh et al., 2020; Kale and Rastogi, 2020; Cheng et al., 2021).



Stephen Chow

Year	Title	Role
1992	Justice, My Foot!	Sung Sai Kit
	Royal Tramp	Wai Siu-bo
	King of Beggars	So Chan

T5 (60.7 BLEU): In 1992, Stephen Chow played Sung Sai Kit in Justice, My Foot!, Wai Siu-bo, King of Beggars and So Chan in Royal Tramp.

LATTICE (87.6 BLEU): In 1992, Stephen Chow appeared as Sung Sai Kit in Justice, My Foot!, Wai Siu-bo in Royal Tramp and King of Beggars as So Chan.

Stephen Chow

Year	1992		
Role	Sung Sai Kit	So Chan	Wai Siu-bo
Title	Justice, My Foot!	King of Beggars	Royal Tramp

T5 (1.3 BLEU): In 1992, Stephen Chow appeared in Royal Tramp, King of Beggars, Justice, My Foot!, Wai Siu-bo, Wai Siu-bo, and as Sung Sai Kit in So Chan.

LATTICE (87.6 BLEU): In 1992, Stephen Chow appeared as Sung Sai Kit in Justice, My Foot!, Wai Siu-bo in Royal Tramp and King of Beggars as So Chan.

Figure 1: Description generation on content-equivalent tables with different layouts by T5 and LATTICE[2]. Correct film-role pairs in generations are in orange. We report also the BLEU-4 score of each generation. T5 is brittle to layout changes, while LATTICE return consistent results.

Prior studies on controlled table-to-text generation often employ a sequence-to-sequence generation method, which merely captures the table as a linear structure (Parikh et al., 2020; Kale and Rastogi, 2020; Su et al., 2021). However, table layouts, though overlooked by prior studies, are key to the generation from two perspectives. First, table layouts indicate the relations among cells that

[1]Code and data will be released after paper publication.

[2]This example is from the ToTTo dataset. The film names and role names in original example is too long. For presentation, we replace the actor name, film names and role names.

collectively present a fact, which are however not simply captured by a linearized table. For example, if we linearize the first table row-wise in Fig. 1, *Wai Siu-bo* will be next to both *Royal Tramp* and *King of Beggers*, so that it is not clear this role belongs to which film. Second, the same content can be equivalently expressed in tables with different layouts, while linearization simplifies the layout representation, it causes brittle generation when table layouts change. Fig. 1 shows two tables with the same content but different layouts, for which the generations by T5 are largely inconsistent.

In this paper, we focus on improving controlled table-to-text generation systems by incorporating two properties: *structure-awareness* and *transformation-invariance*. Structure-awareness, which seeks to understand cell relations indicated by the table structure, is essential for capturing contextualized cell information. Transformation-invariance, which seeks to make the model insensitive to content-invariant structural transformations (including transpose, row shuffle and column shuffle), is essential for model robustness. However, incorporating structure-awareness and transformation-invariance into existing generative neural networks is nontrivial, especially when preserving the generation ability of pretrained models as much as possible.

We enforce the awareness of table layouts and robustness to content-invariant structural transformations on pretrained generative models with an equivariance-learning framework, namely Layout Aware and TransformaTion Invariant Controlled Table-to-Text GEneration (LATTICE). LATTICE encodes tables with a transformation-invariant graph masking technology. This prunes the full self-attention structure into an order-invariant graph-based attention that captures the connected graph of cells belonging to the same row or column, and differentiates between relevant cells and irrelevant cells from the structural perspective. LATTICE also modifies the positional encoding mechanism to preserve the relative position of tokens within the same cell but enforces position invariance among different cells. Our technology is free to be plugged into existing table-to-text generation models, and has improved T5-based models (Raffel et al., 2020) on ToTTo (Parikh et al., 2020) and HiTab (Cheng et al., 2021). Moreover, on a harder version of ToTTo, we preserve promising performance, while previous SOTA systems, even with transformation-based data augmentation, have seen significant performance drops.

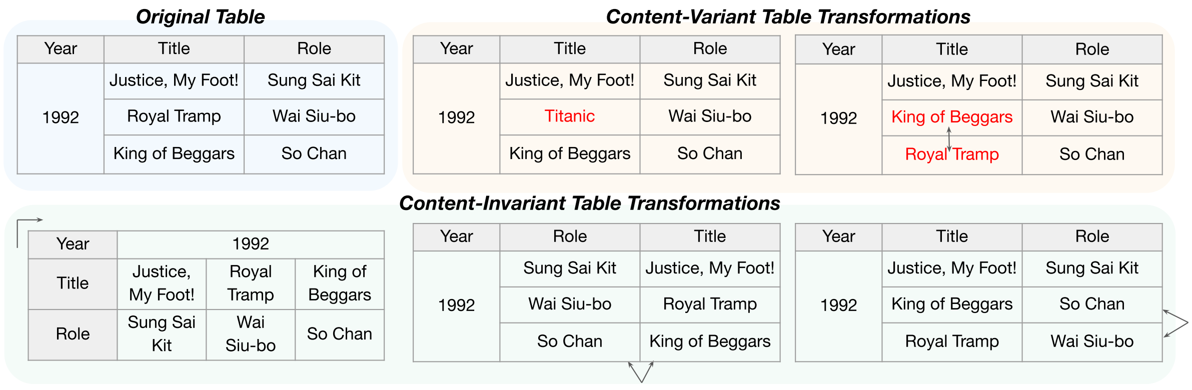
Our contributions are three-fold. First, we propose two essential properties of a precise and robust controlled table-to-text generation system, i.e. structure-awareness and transformation-invariance. Second, we demonstrate how our transformation-invariant graph masking technology can enforce these two properties, and effectively enhance a representative group of Transformer-based generative models, i.e. T5-based models, for more generalizable and accurate generation. Third, in addition to experiments on ToTTo and HiTab benchmarks, we evaluate our model on a harder version of ToTTo with a special focus on robustness to content-invariant structural transformations.

2 Method

In this section, we first describe the preliminaries of content-invariant table transformations, base models and the input format for controlled table-to-text generation (§2.1). Then we introduce the technical details about how the transformation-invariant graph masking technology in LATTICE enforces the model to be structure-aware and transformation-invariant (§2.2). Finally, we present two alternative techniques for strengthening the transformation-invariance to be compared with LATTICE (§2.3).

2.1 Preliminaries

Content-Invariant Table Transformations. Tables organize and present information by row and column. A piece of information is presented in a cell (with headers), which is the basic unit of a table. Rows and columns are high-level units, indicating relations among cells and combining them to express more comprehensive information. We discuss two categories of transformations that may be made on a table, as shown in Fig. 2. First, *content-variant* transformations modify or exchange a part of cells in different rows or columns changing the semantics of the table. In such cases, new tabular content are created including information being inconsistent with the original table. Second, *content-invariant* transformations consists of operations that do not influence content within combinations of the same rows and columns, resulting in semantically equal (sub-)tables. Specifically, such operations include transpose, row shuffle and column shuffle. By performing any or a combination of such operations, we can present the same infor-


Original Table

Year	Title	Role
1992	Justice, My Foot!	Sung Sai Kit
	Royal Tramp	Wai Siu-bo
	King of Beggars	So Chan

Content-Variant Table Transformations

Year	Title	Role
1992	Justice, My Foot!	Sung Sai Kit
	Titanic	Wai Siu-bo
	King of Beggars	So Chan

Year	Title	Role
1992	Justice, My Foot!	Sung Sai Kit
	King of Beggars	Wai Siu-bo
	Royal Tramp	So Chan

Content-Invariant Table Transformations

Year	1992		
Title	Justice, My Foot!	Royal Tramp	King of Beggars
Role	Sung Sai Kit	Wai Siu-bo	So Chan

Year	Role	Title
1992	Sung Sai Kit	Justice, My Foot!
	Wai Siu-bo	Royal Tramp
	So Chan	King of Beggars

Year	Title	Role
1992	Justice, My Foot!	Sung Sai Kit
	King of Beggars	So Chan
	Royal Tramp	Wai Siu-bo

Figure 2: Examples of different types of table transformations. Arrows indicate how specific operations change the positions of tables components. Modifications causing semantic changes are in red.

mation in different table layouts.

Base Models. Pretrained Transformer-based generative models achieve SOTA performance on various text generation tasks (Raffel et al., 2020; Lewis et al., 2020). In order to adapt this kind of models to table-to-text generation, prior works propose to linearize the table into a textual sequence (Kale and Rastogi, 2020; Chen et al., 2020b; Su et al., 2021). Our method LATTICE is model-agnostic and can be incorporated into any such models. Following Kale and Rastogi (2020), we choose a family of the best performing models, T5 (Raffel et al., 2020), as our base models. Models of this family are jointly pretrained on a series of supervised and self-supervised text-to-text tasks. Models can switch between different tasks by prepending a task-specific prefix to the input. Our experiments (§3.3 and §3.4) point out that base models are brittle to content-invariant table transformations and can only capture limited layout information.

Input Format. Prior works (Kale and Rastogi, 2020; Chen et al., 2020b; Su et al., 2021) linearize (highlighted) table cells based on row and column indexes. The input sequence often starts with the metadata of a table, such as page title and section title. Then, it traverses the table row-wise from the top-left cell to the bottom-right cell. Headers of each cell can be either treated as individual cells or appended to the cell content. Each metadata/cell/header field is separated with special tokens. This linearization process suits the input to text-to-text generation models, yet discards much of the structural information of a table (e.g., two cells in the same column can be separated by irrelevant cells in the sequence, while the last cell and first cell in adjacent rows can be adjacent although they are irrelevant), and is sensitive to content-invariant table transformations.

2.2 Transformation-Invariant Graph Masking

LATTICE realizes equivariance learning by modifying the Transformer encoder architecture. It also improves the base model's ability of capturing structures of highlighted tabular content. Specifically, we incorporate a structure-aware self-attention mechanism and a transformation invariant positional encoding mechanism in the base model The workflow is shown in Fig. 3.

Structure-Aware Self-Attention. Transformer (Vaswani et al., 2017) adopts self-attention to aggregate information from all the tokens in the input sequence. The attention flows form a complete graph connecting each token. This mechanism works well for modeling sequences but falls short of capturing tabular structures. The non-linear layout structure reflects semantic relations among cells, hence should be captured by self-attention.

We incorporate structural information by pruning the attention flows. According to the nature of information arrangement in a table, two cells in neither the same row nor the same column are not semantically related, or at least the combination of them do not directly express information this table seeks to convey. Intuitively, representations of these cells should not directly pass information to each other. In LATTICE, attention flows among tokens of semantically unrelated cells are removed from the attention graph, while those within the metadata, within each cell, and between metadata and each cell are preserved. In this way, we also ensure the transformation-invariance property of the self-attention mechanism as related cells in the same row or same column are all linked in an unordered way in the attention graph. It is easy to show that for any individual cell, the links in

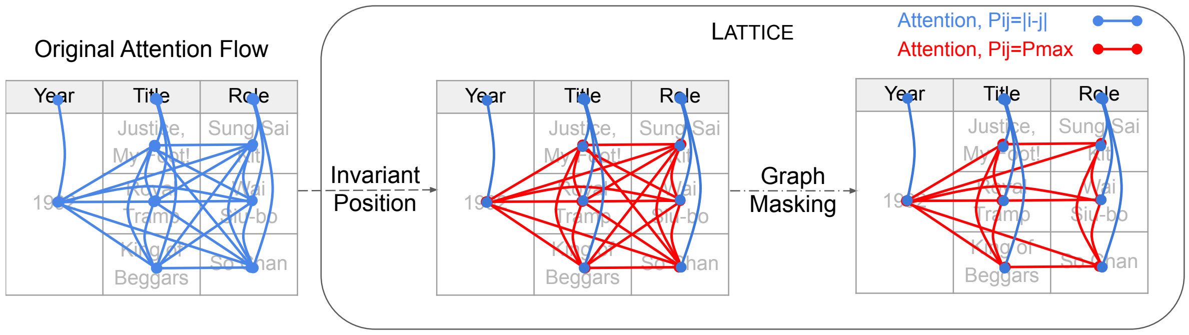


Figure 3: Attention flows of the base model and LATTICE. In this example, we adopt the input format which appends headers to each cell, so headers can be seen as part of the cell content. We omit the attention flows among tokens within a cell, as they are in the same type of the flows between headers and corresponding cells. P_{ij} represents the relative position between tokens at both ends of the attention flow, where i and j are absolute positions of tokens in the linearized table and P_{max} is the max relative position allowed. The base model has a complete attention graph among all cells with relative positions based on linear distance. LATTICE prunes the attention flow based on the table layout and assigns transformation-invariant relative positions between cells.

the attention graph will remain the same after any content-invariant operations (§2.1) are applied.

Transformation-Invariant Positional Encoding. When calculating the attention scores between each pair of tokens, the base model captures their relative position in the sequence of linearized table as an influential feature. Specifically, the attention flow from the i-th token to the j-th token is paired with a relative position $P_{ij} = |i - j|$. This easily causes positional biases among distinct cells, since the relative positions in the sequence do not fully reflect relations among cells in the table. Moreover, the relative position between the same token pair will change as the table layout change, which is the source of inconsistent generation shown in Fig. 1.

As discussed in §2.1, for a given cell, its relations with other cells in the same row or column should be equally considered. It is natural to assign the same relative positions among (tokens of) cells in the same row or column, no matter how far their distance is in the linear sequence. Meanwhile, we preserve the relative positions of tokens inside the same cell (or the metadata). Specifically, the relative position between the i-th token and the j-th token in the input sequence is

$$P_{ij} = P_{ji} = \begin{cases} |i-j|, & \text{if in the same field;} \\ P_{max}, & \text{otherwise;} \end{cases}$$

where "same field" means the two tokens are from the same cell or both of them are from the metadata, and P_{max} is the max relative position allowed. As a result, LATTICE represents cells (and the metadata) in a way that is invariant to their relative positions in the sequence. As content-invariant table transformations do not change the relations among cells in the table (i.e. whether two cells are from the same row or column), this positional encoding mechanism is transformation-invariant.

Training and Inference. After obtaining the structure-aware and transformation-invariant table representation, LATTICE conducts similar training and inference as the base model. Given the linearized table T_i, its layout structure S_i, and target sentence $Y_i = \{y_1^i, y_2^i, ..., y_{n_i}^i\}$, training minimizes the negative log-likelihood. For a dataset (or batch) with N samples, the loss function is

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{n_i} \log P(y_j^i | y_{<j}^i, T_i, S_i).$$

During inference, the model generates a sentence token by token, where each time it outputs a distribution over a vocabulary.

2.3 Alternative Techniques

In addition to the equivariance learning realized by tranformation-invariant graph masking, we present and compare with two alternative techniques.

Layout-Agnostic Input. The first technique is adjusting input sequences to be invariant to content-invariant table transformations. A simple way is to reorder cells and headers by an arbitrary order not based on table layouts (e.g., lexicographic order) to form a sequence. If any special tokens are used to separate cells and headers, they should also include no layout information[3]. As a result, this input format loses all information about table layouts.

[3]For example, use <*header*> instead of <*row_header*>.

Data Augmentation. The second technique is data augmentation by content-invariant table transformation. This technique augment tables with different layouts to training data, seeking to enhance the robustness of the base model by exposing it to more diverse training instances.

Our experiments systematically compares these two techniques with tranformation-invariant graph masking in §3.3, revealing how directly performing equivariance learning from the perspective of neural network structure leads to better performance and robustness than using layout-agnostic input or data augmentation.

3 Experiments

In this section, we conduct experiments on two benchmark datasets. First, we introduce the details of datasets, baselines, evaluation metrics and our implementation (§3.1). Then, we show the overall performance of LATTICE (§3.2). After that, we analyze the model robustness on a harder version of the ToTTo dataset where content-invariant perturbations are introduced (§3.3). Finally, we provide ablation study on components of transformation-invariant graph masking (§3.4).

3.1 Experimental Settings

Datasets. We evaluate our model on ToTTo (Parikh et al., 2020) and HiTab (Cheng et al., 2021) benchmarks. Details of them are described as follows (see Appx. §A for more information):

- **ToTTo:** ToTTo consists of 83,141 Wikipedia tables, 120,761/7,700/7,700 sentences (i.e. descriptions of tabular data) for train/dev/test. Target sentences in test set are not publicly available. Each sentence is paired with a set of highlighted cells in a table, and each table has metadata including its page title and section title. The dev and test sets can be further split into 2 subsets, i.e. overlap and non-overlap, according to whether the table exists in the train set.
- **HiTab:** HiTab contains 3,597 tables, including tables from statistical reports and Wikipedia, forming 10,686 samples distributed across train (70%), dev (15%), and test (15%). Each sample consists of a target sentence and a table with highlighted cells and hierarchical headers.

Evaluation Metrics. We adopt three widely used evaluation metrics for text generation. BLEU (Papineni et al., 2002) is one of the most common metric for text generation based on n-gram co-occurrence. We use the commonly used BLEU-4 following prior works (Parikh et al., 2020; Cheng et al., 2021). PARENT (Dhingra et al., 2019) is a metric for data-to-text evaluation taking both references and tables into account. BLEURT (Sellam et al., 2020) is a learned evaluation metric for text generation based on BERT (Devlin et al., 2019). Following prior studies (Parikh et al., 2020; Cheng et al., 2021), we report all three metrics on ToTTo and the first two metrics on HiTab using the evaluation tool released by Parikh et al. (2020).

Baselines. We present baseline results of the following representative methods:

- **Pointer-Generator** (Gehrmann et al., 2018): An LSTM-based encoder-decoder model with attention and copy mechanism, first proposed by See et al. (2017) for text summarization.
- **BERT-to-BERT** (Rothe et al., 2020): A Transformer-based encoder-decoder model, where the encoder and decoder are initialized with BERT (Devlin et al., 2019).
- **T5** (Kale and Rastogi, 2020): A pretrained generation model first proposed by Raffel et al. (2020). The model is Transformer-based, pretrained on text-to-text tasks, and finetuned on linearized tables to offer the previous SOTA performance.

All the baseline results on ToTTo can be found in the official leaderboard[4], except for T5-small and T5-base, for which we reproduce the results on dev set reported by Kale and Rastogi (2020) and submit the predictions on hidden test set to the leaderboard. For HiTab, we run T5 and LATTICE using our replication of the linearization process introduced by Cheng et al. (2021)[5]. Results of other baselines are from Cheng et al. (2021).

Implementation Details. We adopt the pretrained model weights released by Raffel et al. (2020). Specifically, we use T5-small and T5-base[6]. For finetuning, we use a batch size of 8 and a constant learning rate of $2e^{-4}$. Following Kale and Rastogi (2020); Cheng et al. (2021), all input sequences

[4]`https://github.com/google-research-datasets/ToTTo`

[5]According to the authors, their linearization process needs unreleased raw excel files. We reproduce it with released tables which results in less precise and informative inputs.

[6]Although a previous study (Kale and Rastogi, 2020) has obtained better results using the much larger T5-3B, we were not able to run that model on our equipment even with a batch size of 1 due to the overly excessive GPU memory usage.

Model	Overall			Overlap			Non-Overlap		
	BLEU	PARENT	BLEURT	BLEU	PARENT	BLEURT	BLEU	PARENT	BLEURT
Pointer-Generator	41.6	51.6	0.076	50.6	58.0	0.244	32.2	45.2	-0.092
BERT-to-BERT	44.0	52.6	0.121	52.7	58.4	0.259	35.1	46.8	-0.017
T5-small	45.3	57.0	0.187	52.7	61.0	0.316	37.8	53.0	0.057
LATTICE (T5-small)	**47.4**	**57.8**	**0.207**	**55.6**	**62.3**	**0.337**	**39.1**	**53.3**	**0.077**
T5-base	47.4	56.4	0.221	55.5	61.1	0.344	39.1	51.7	0.098
LATTICE (T5-base)	**48.4**	**58.1**	**0.222**	**56.1**	**62.4**	**0.345**	**40.4**	**53.9**	**0.099**

Table 1: Results on ToTTo test set. Best scores are in bold.

Model	BLEU	PARENT
Pointer-Generator	5.8	8.8
BERT-to-BERT	11.4	16.7
T5-small	14.2	22.0
LATTICE (T5-small)	**15.7**	**23.8**
T5-base	14.7	21.9
LATTICE (T5-base)	**16.3**	**22.7**

Table 2: Results on HiTab test set.

are truncated to a length of 512 to accommodate the limit of the pretrained models. Although our model can achieve consistent performance with any input format, we adopt the layout-agnostic input format (§2.3) to avoid uncertainty due to truncation and special markers. More details about our implementation are in Appx. §B.

3.2 Main Results

Tab. 1 shows model performance on ToTTo test set. Among the baselines, methods based on pretrained Transformer models (i.e. BERT-to-BERT and T5) outperform the others and T5 models perform the best. Our method LATTICE can be plugged into such models. We compare our method with pure T5 models of different sizes, and LATTICE consistently performs better. Overall, LATTICE (T5-small) achieves improvements of 2.1 BLEU points and 0.8 PARENT points in comparison with T5-small, and LATTICE (T5-base) achieves improvements of 1.0 BLEU points and 1.7 PARENT points in comparison with T5-base. These results indicate the importance of structure information, which is almost totally abandoned by baselines. Further, the performance gain on tables both seen and unseen during training are significant. Specifically, on the overlap subset, LATTICE (T5-small) achieves improvements of 2.9 BLEU points and 1.3 PARENT points, and LATTICE (T5-base) achieves improvements of 0.9 BLEU points and 1.3 PARENT points, indicating better intrinsic performance. On the non-overlap subset, LATTICE (T5-small) achieves improvements of 1.3 BLEU points and 1.0 PARENT points, and LATTICE (T5-base) achieves improvements of 1.3 BLEU points and 2.2 PARENT points, indicating LATTICE is more generalizable to unseen tables. We also observe that the improvement on BLEURT is not as much as the other two metrics. It is reasonable as BLEURT is trained with machine translation annotations and synthetic data by mask filling, backtranslation and word drop. These training data ensures its robustness to surface generation but not reasoning-based generation. Although the effectiveness of BLEURT is verified on an RDF-to-text dataset, tabular data holds different properties with RDF data[7].

Results on HiTab in Tab. 2 further verify the effectiveness and generalizability of LATTICE. For different model sizes, LATTICE consistently performs better than T5 models. We also observe that on this dataset the model with highest BLEU score is not the model with highest PARENT score. It is partially because of the annotations. Many numbers appear in both tables and target sentences are of different precision. Copying such numbers from tables to generated sentences may increase PARENT score but reduce BLEU score.

3.3 Robustness Evaluation

To further evaluate model robustness against content-invariant perturbations on tables, we create a harder version of the ToTTo dev set, where each table is perturbed with a combination of row-wise shuffling, column-wise shuffling and table transpose. Especially, models can no longer benefit from memorizing the layout of tables appearing in both train set and dev set. We compare four methods based on T5, including the basic version proposed by Kale and Rastogi (2020), enhanced T5 with the layout-agnostic input or data augmentation (§2.3), and T5 incorporated in LATTICE.

According to the results shown in Tab. 3, vanilla T5 models face a severe performance drop when content-invariant perturbations are introduced. Overall, BLEU scores drop by 3.4 for T5-small, and 4.5 for T5-base. We also observe that

[7]For example, in the ToTTo dataset, 21% samples requires reasoning while 13% samples requires comparison.

Model	Overall			Overlap			Non-Overlap		
	Origin	Transform	Δ	Origin	Transform	Δ	Origin	Transform	Δ
T5-small	45.7	42.3	-3.4	53.7	49.3	-4.4	37.7	35.4	-2.3
+ layout-agnostic input	44.2	44.2	**0**	51.6	51.6	**0**	37.0	37.0	**0**
+ data augmentation	45.3	44.4	-0.9	52.8	52.0	-0.8	37.9	37.0	-0.9
LATTICE (T5-small)	**47.5**	**47.5**	**0**	**55.5**	**55.5**	**0**	**39.5**	**39.5**	**0**
T5-base	47.4	42.9	-4.5	55.8	50.7	-5.1	39.2	35.4	-3.8
+ layout-agnostic input	46.2	46.2	**0**	54.3	54.3	**0**	38.3	38.3	**0**
+ data augmentation	47.2	46.9	-0.3	55.3	54.8	-0.5	39.2	38.9	-0.3
LATTICE (T5-base)	**48.6**	**48.6**	**0**	**56.6**	**56.6**	**0**	**40.8**	**40.8**	**0**

Table 3: Robustness evaluation on ToTTo dev set. *Origin* is the BLUE score on original tables, while *Transform* is the BLUE score on transformed tables. All transformed tables are transposed, row shuffled and column shuffled. Δ is the difference between the two scores. Best scores in each group are in bold.

Att	**Pos**	Overall	Overlap	Non-Overlap
-	-	45.7	53.7	37.7
✓	-	47.0	54.4	39.6
✓	✓	47.5	55.5	39.5

Table 4: Ablation study on ToTTo dev set. Scores are BLEU. *Att* and *Pos* denote structure-aware self-attention and transformation-invariant positional encoding.

the performance drop on overlap subset is larger than on non-overlap subset. This indicates that the performance gain of T5 models is somehow due to their memory of some tables existing in train set, which is however brittle and not generalizable. Applying layout-agnostic input format, which linearizes tables by lexicographic order instead of cell index order, ensures models to return stable predictions, but results in worse overall performance due to the loss of structural information. Not surprisingly, layout-agnostic input causes performance drops by 1.5 BLEU points and 1.2 BLEU points to T5-small and T5-base on original dev set.

Another common way to improve model robustness is increasing the diversity of training instances with data augmentation. We augment the original training set by 8-fold using the three content-invariant transformation operations and their combinations. Training with augmented data reduces the gap between model performance on original tables and transformed tables. However, data augmentation is never exhaustive enough to guarantee true equivariance. Also, this introduces different variants of the same table into the train set, so there is a gap between the same table in train set and dev set. As a result, the performance on overlap subset is slightly worse than without data augmentation, but the performance on non-overlap subset is not negatively influenced. LATTICE guarantees consistent predictions towards content-invariant table transformations while achieving the best performance. In comparison with using layout-agnostic input format which also guarantees equivariance, LATTICE (T5-small) provides additional 3.3 BLEU points, and LATTICE (T5-base) provides additional 2.4 BLEU points on original dev set.

3.4 Ablation Study

To help understand the effect of two key mechanisms in transformation-invariant graph masking, we hereby present ablation study results in Tab. 4.

Structure-Aware Self-Attention. We examine the effectiveness of structure-aware self-attention. In comparison with original (fully-connected) self-attention, incorporating structural information by pruning attention flows can improve the overall performance by 1.3 BLEU points. Detailed scores on two subsets show that both tables seen and unseen during training can benefit from structural information. The consistent improvements on two subsets indicate that structure-aware self-attention improves model ability of capturing cell relations rather than memorizing tables.

Transformation-Invariant Positional Encoding. We further test the effectiveness of transformation-invariant positional encoding. We observe that although this technique is mainly designed for ensuring model robustness towards layout changes, it can bring an additional improvement of 0.5 BLEU points to overall performance. Interestingly, the improvement is mainly on the overlap subset. We attribute it to the fact that the same table in train set and dev set may have different highlighted cells, so that memorizing the layout information in train set hinders in-domain generalization.

4 Related Work

We review two relevant research topics. Since both topics have a large body of work, we provide a selected summary.

Table-to-text Generation. Table-to-text generation seeks to generate textual descriptions for tabular data. In comparison to text-to-text generation, the input of table-to-text generation is semi-structured data. Early studies adapt the encoder-decoder framework to data-to-text generation with encoders aggregating cell information (Lebret et al., 2016; Wiseman et al., 2017; Bao et al., 2018). Followed by the success of massively pre-trained sequence-to-sequence Transformer models (Raffel et al., 2020; Lewis et al., 2020), recent SOTA systems apply these models to table-to-text generation (Kale and Rastogi, 2020; Su et al., 2021), where the input table is linearized to a textual sequence.

A table can include ample information and it is not always able to be summarized in one sentence. A line of work learns to generate selective descriptions by paying attention to key information in the table (Perez-Beltrachini and Lapata, 2018; Ma et al., 2019). However, multiple statements can be entailed from a table when different parts of the table are focused on. To bridge this gap, Parikh et al. (2020) proposes controlled table-to-text generation, allowing the generation process to react differently according to distinct highlighted cells. As highlighted cells can be at any positions and of arbitrary numbers, simple linearization, which breaks the layout structure, hinders relations among cells from being captured, therefore causing unreliable or hallucinated descriptions to be generated.

A few prior studies introduce structural information to improve model performance on table-to-text generation, either by incorporating token position (Liu et al., 2018), or by aggregating row and column level information (Bao et al., 2018; Nema et al., 2018; Jain et al., 2018). However, none of existing methods can be directly applied to pretrained Transformer-based generative models, especially when we want to ensure model robustness to content-invariant table transformations. Our method enforces both structure-awareness and transformation-invariance to such models.

Equivariant Representation Learning. Equivariance is a type of prior knowledge existing broadly in real-world tasks. Earlier studies show that incorporating equivariance learning can improve visual perception model robustness against turbulence caused by geometric transformations, such as realizing translation, rotation, and scale equivariance of images (Lenc and Vedaldi, 2015; Worrall et al., 2017; Ravanbakhsh et al., 2017; Sosnovik et al., 2019; Yang et al., 2020). The input to those tasks presents unstructured information, and several geometrically invariable operations are incorporated in neural networks to realize the aforementioned equivariance properties. For example, Convolutional Neural Networks (CNNs) are equivariant to translations in nature (Lenc and Vedaldi, 2015). Harmonic Networks and Spherical CNNs extend the equivariance of CNNs to rotations (Worrall et al., 2017; Esteves et al., 2018). Group Equivariant Convolutional Networks are equivariant to more spatial transformations including translations, rotations and reflections (Cohen and Welling, 2016). Nonetheless, none of these geometrically invariable techniques can be directly applied to Transformer-based generative models to ensure equivariance on (a part of) structured tabular data, which is exactly the focus of this work. Our method realizes equivariant intermediate representations against content-invariant table transformations in table-to-text generation.

Some other works, while not explicitly using equivariant model structures, seek to realize equivariant representations by augmenting more diverse changes into training data (Chen et al., 2020a; Wu et al., 2020). Although the model can benefit from seeing more diverse inputs involving content-invariant transformations (Wu et al., 2020), this strategy has two drawbacks. The augmented data, while introducing much computational overhead to training, are never exhaustive enough to guarantee true equivariance. By contrast, our method guarantees equivariance through the neural network design and do not introduce much training overhead.

5 Conclusion

We propose LATTICE, a structure-aware equivariance learning framework for controlled table-to-text generation. Our experimental results verify the importance of structure-awareness and transformation-invariance, two key properties enforced in LATTICE, towards precise and robust description generation for tabular content. The proposed properties and equivariance learning framework aligns well with the nature of information organized in tables. Future research can consider extending the structure-aware equivariance learning framework to other data-to-text generation tasks (Koncel-Kedziorski et al., 2019; Nan et al., 2021), and tabular reasoning or retrieval tasks (Gupta et al., 2020; Wang et al., 2021a,b).

Ethical Considerations

This work seeks to develop a structure-aware equivariance learning framework for table-to-text generation. Since the proposed method focuses on improving prior generation systems by better utilization of structural information, it does not introduce bias towards specific content. The distinction between beneficial use and harmful use depends mainly on the data. Proper use of the technology requires that input corpora are legally and ethically obtained. We conduct experiments on two open benchmark in the way they intended to. Although we create a harder version of ToTTo dev set, the table transformation operations we use are content-invariant, whereas the ground-truth generation remains the same as it is in the original dataset, ensuring no further social bias is introduced.

References

Junwei Bao, Duyu Tang, Nan Duan, Zhao Yan, Yuanhua Lv, Ming Zhou, and Tiejun Zhao. 2018. Table-to-text: Describing table region with natural language. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Shuxiao Chen, Edgar Dobriban, and Jane H Lee. 2020a. A group-theoretic framework for data augmentation. *Journal of Machine Learning Research*, 21(245):1–71.

Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020b. Logical natural language generation from open-domain tables. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7929–7942.

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. In *International Conference on Learning Representations*.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. 2020c. Hybridqa: A dataset of multi-hop question answering over tabular and textual data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 1026–1036.

Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. 2021. Hitab: A hierarchical table dataset for question answering and natural language generation. *arXiv preprint arXiv:2108.06712*.

Taco Cohen and Max Welling. 2016. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR.

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.

Bhuwan Dhingra, Manaal Faruqui, Ankur Parikh, Ming-Wei Chang, Dipanjan Das, and William Cohen. 2019. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4884–4895.

Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. 2018. Learning so(3) equivariant representations with spherical cnns. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–68.

Sebastian Gehrmann, Falcon Dai, Henry Elder, and Alexander M Rush. 2018. End-to-end content and plan selection for data-to-text generation. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 46–56.

Vivek Gupta, Maitrey Mehta, Pegah Nokhiz, and Vivek Srikumar. 2020. Infotabs: Inference on tables as semi-structured data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2309–2324.

Parag Jain, Anirban Laha, Karthik Sankaranarayanan, Preksha Nema, Mitesh M Khapra, and Shreyas Shetty. 2018. A mixed hierarchical attention based encoder-decoder approach for standard table summarization. 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 622–627.

Mihir Kale and Abhinav Rastogi. 2020. Text-to-text pre-training for data-to-text tasks. In *Proceedings of the 13th International Conference on Natural Language Generation*, pages 97–102.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text generation from knowledge graphs with graph transformers. 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 2284–2293.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213.

Karel Lenc and Andrea Vedaldi. 2015. Understanding image representations by measuring their equivariance and equivalence. In *Proceedings of the IEEE* *conference on computer vision and pattern recognition*, pages 991–999.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. 2018. Table-to-text generation by structure-aware seq2seq learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Shuming Ma, Pengcheng Yang, Tianyu Liu, Peng Li, Jie Zhou, and Xu Sun. 2019. Key fact as pivot: A two-stage model for low resource table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2047–2057.

Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, et al. 2021. Dart: Open-domain structured data record to text generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 432–447.

Preksha Nema, Shreyas Shetty, Parag Jain, Anirban Laha, Karthik Sankaranarayanan, and Mitesh M Khapra. 2018. Generating descriptions from structured data using a bifocal attention mechanism and gated orthogonalization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1539–1550.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Ankur P Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of EMNLP*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037.

Laura Perez-Beltrachini and Mirella Lapata. 2018. Bootstrapping generators from noisy data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1516–1527.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Siamak Ravanbakhsh, Jeff Schneider, and Barnabas Poczos. 2017. Equivariance through parameter-sharing. In *International Conference on Machine Learning*, pages 2892–2901. PMLR.

Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. Bleurt: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892.

Ivan Sosnovik, Michał Szmaja, and Arnold Smeulders. 2019. Scale-equivariant steerable networks. In *International Conference on Learning Representations*.

Yixuan Su, David Vandyke, Sihui Wang, Yimai Fang, and Nigel Collier. 2021. Plan-then-generate: Controlled data-to-text generation via planning. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 895–909.

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.

Fei Wang, Kexuan Sun, Muhao Chen, Jay Pujara, and Pedro Szekely. 2021a. Retrieving complex tables with multi-granular graph representation learning. In *SIGIR*, page 1472–1482.

Fei Wang, Kexuan Sun, Jay Pujara, Pedro Szekely, and Muhao Chen. 2021b. Table-based fact verification with salience-aware learning. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4025–4036.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263.

Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

Daniel E Worrall, Stephan J Garbin, Daniyar Turmukhambetov, and Gabriel J Brostow. 2017. Harmonic networks: Deep translation and rotation equivariance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5028–5037.

Sen Wu, Hongyang Zhang, Gregory Valiant, and Christopher Ré. 2020. On the generalization effects of linear transformations in data augmentation. In *International Conference on Machine Learning*, pages 10410–10420. PMLR.

Qin Yang, Chenglin Li, Wenrui Dai, Junni Zou, Guo-Jun Qi, and Hongkai Xiong. 2020. Rotation equivariant graph convolutional network for spherical image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4303–4312.

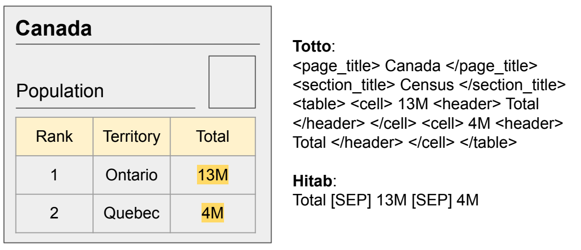
Appendices

A Dataset Profile

ToTTo. An English dataset released under the Apache License v2.0. The dataset is dedicated for controlled table-to-text generation.

HiTab. An English dataset released under Microsoft's Computational Use of Data Agreement (C-UDA). It is intended for both controlled table-to-text generation and table QA with a special focus on hierarchical tables.

B Implementation Details



Rank	Territory	Total
1	Ontario	13M
2	Quebec	4M

Totto:
```
<page_title> Canada </page_title>
<section_title> Census </section_title>
<table> <cell> 13M <header> Total
</header> </cell> <cell> 4M <header>
Total </header> </cell> </table>
```

Hitab:
```
Total [SEP] 13M [SEP] 4M
```

Figure 4: Example to demonstrate input formats. Highlighted cells are *13M* and *4M*.

Input Format. As shown in Fig. 4, we use different input formats for ToTTo and Hitab following prior works (Kale and Rastogi, 2020; Cheng et al., 2021), since the tables and annotations in these two datasets have different properties.

For ToTTo, we follow the linearization procedure of Kale and Rastogi (2020). Specifically, the textual sequence consists of the page title, section title, table headers and cells. Each cell may be associated with multiple row and column headers. Special markers are used to denote the begin and end of each field. Different from Kale and Rastogi (2020), we use the same markers for row headers and column headers.

For HiTab, we follow the linearization procedure of Cheng et al. (2021). Specifically, the textual sequence consists of highlighted cells and headers, headers of highlighted cells, and cells belong to highlighted headers. A universal separator token `[SEP]` is used.

Model Details. LATTICE does not add any parameters to the base model, so LATTICE (T5-small) has 60 million parameters and LATTICE (T5-base) has 220 million parameters, same as the base models. For ToTTo, we use a beam size of 4 to generate sentences with at most 128 tokens. For HiTab, we use a beam size of 5 to generate sentences with at most 60 tokens following Cheng et al. (2021). Our implementation is based on Pytorch (Paszke et al., 2019) and Transformers (Wolf et al., 2020). We run experiments on a commodity server with a GeForce RTX 2080 GPU. It takes about 0.5 hour to train LATTICE (T5-small) for 10,000 steps and about 1 hour to train LATTICE (T5-base) for 10,000 steps. Considering different sizes of two datasets, we train models for 150,000 steps on ToTTo, and for 20,000 steps on HiTab. Results of LATTICE on ToTTo dev set and HiTab are average of multiple runs. For ToTTo test set, we report the results on official leaderboard.