LOFT: Enhancing Faithfulness and Diversity for Table-to-Text Generation via Logic Form Control

Yilun Zhao∗1  Zhenting Qi2  Linyong Nan1
Lorenzo Jaime Yu Flores1  Dragomir Radev1
1Yale University  2Zhejiang University
yilun.zhao@yale.edu  zhenting.19@intl.zju.edu.cn

Abstract

Logical Table-to-Text (LT2T) generation is tasked with generating logically faithful sentences from tables. There currently exists two challenges in the field: 1) Faithfulness: how to generate sentences that are factually correct given the table content; 2) Diversity: how to generate multiple sentences that offer different perspectives on the table. This work proposes LOFT, which utilizes logic forms as fact verifiers and content planners to control LT2T generation. Experimental results on the LOGICNLG dataset demonstrate that LOFT is the first model that addresses unfaithfulness and lack of diversity issues simultaneously. Our code is publicly available at https://github.com/Yale-LILY/LoFT.

1 Introduction

Table-to-Text (T2T) generation aims to produce natural language descriptions from structured tables. A statement generated from tabular data can be inferred based on different levels of information (e.g., value of a specific cell, logical operation result across multiple cells). Although current T2T models (Lebret et al., 2016; Wiseman et al., 2017; Puduppully et al., 2019; Parikh et al., 2020) have shown remarkable progress in fluency and coherence, they mainly focus on surface-level realizations without much logical inference.

Recently, Chen et al. (2020a) proposed LOGICNLG, which is tasked with generating textual descriptions that require logical reasoning over tabular data (i.e., LT2T generation). LT2T generation is challenging as it requires a model to learn the logical inference knowledge from table-text pairs and generate multiple factually correct sentences. Another challenge for LT2T generation is the diversity of generated text. Natural Language Generation (NLG) encourages the diverse output of statements over a single input, as it provides various perspectives on the data and offers users more choices. In LT2T generation, requirements for diversity naturally emerge from the need to apply different logical operations to extract different levels of table information. However, current methods (Chen et al., 2021; Nan et al., 2022; Liu et al., 2022a; Zhao et al., 2022b) that address issues of unfaithfulness have overlooked the importance of diversity. As shown in Figure 1, multiple statements generated using current methods (Nan et al., 2022) might only cover information from the same...
table region or logical operation. Such issues related to lack of diversity could limit the deployment of LT2T models in the real world.

In this work, we attribute unfaithfulness and lack of diversity to the absence of controllability over generation. Specifically, due to the large number of combinations of different logical operations and table regions, the space of factually correct statements is exponentially large. However, LOGIC-NLG uses the whole table as the input, without providing annotations related to any other explicit control attribute. As a result, it is hard and uncontrollable for neural models to decide a favorable choice of logical selections solely based on the table input. We believe such uncontrollability leads to unfaithfulness and lack of diversity issues.

This work proposes LoFT, a framework that utilizes logic forms as mediators to enable controllable LT2T generation. Logic forms (Chen et al., 2020d,b) are widely used to retrieve evidence and explain the reasons behind table fact verification (Yang et al., 2020; Yang and Zhu, 2021; Ou and Liu, 2022). In this work, logic forms are used as: 1) fact verifiers to ensure the factual correctness of each generated sentence; and 2) content planners to control which logical operation and table region to use during the generation. Experimental results show that LoFT surpasses previous methods in faithfulness and diversity simultaneously.

2 Related Work

Logical Table-to-Text (LT2T) Generation

LOGIC-NLG (Chen et al., 2020a) is tasked with generating logically faithful sentences from tables. To improve the faithfulness of generated statements, Nan et al. (2022) trained a system both as a generator and a faithfulness discriminator with additional replacement detection and unlielihood learning tasks. Liu et al. (2022a) pre-trained a model on a synthetic corpus of table-to-logic-form generation. Zhao et al. (2022b) demonstrated that faithfulness of LT2T can be improved by pre-training a generative language model over synthetic Table QA examples. However, these methods overlook the importance of diversity in T2T generation, and might generate multiple statements that cover the same table regions or reasoning operations. Previous methods in NLG proposed to improve diversity by modifying the decoding techniques (Li et al., 2016). However, these approaches degrade faithfulness as measured against baselines (Perlitz et al., 2022). To enable controllable generation and improve diversity, Perlitz et al. (2022) used logical types of statements as a control. However, such methods still suffer from problems related to unfaithfulness, and may generate statements covering limited table regions. This work proposes to leverage the logic form as a fact checker and content planner to control LT2T generation, which tackles the challenges about faithfulness and diversity at the same time.

Table Fact Verification via Logic Form

Logic forms are widely used in Table Fact Verification (Chen et al., 2020b). Specifically, given an input statement, the model (Yang et al., 2020; Yang and Zhu, 2021; Ou and Liu, 2022) will first translate it into logic form. Then the logic form will be executed over the table, and return true/false as the entailment label for a given statement. While several works (Chen et al., 2020d; Shu et al., 2021; Liu et al., 2021) focused on generating fluent statements from logic forms, the utilization of logic forms to benefit LT2T generation is still unexplored.

3 LoFT

This section first introduces the logic form utilized, and then delves into the training and inference process of LoFT. We also explain how the use of logic forms can enhance both faithfulness and text-diversity in LT2T generation.

3.1 Logic Form Implementation

Logic forms are widely used to retrieve evidence and explain the reasons behind table fact verification. We use the same implementation as Chen et al. (2020d), which covers 8 types of the most common logical operations (e.g., count, aggregation) to describe a structured table. Each logical operation corresponds to several Python-based functions. For example, the definition of function all_greater(view, header, value) under “majority” category is: checking whether all the values under header column are greater than value, with the scope (i.e., view) of all or a subset of table rows. The complete list of logical operation types and corresponding function definitions are shown in Table 4 in Appendix.

3.2 LoFT Training

Training Task Formulation

Given the serialized tabular data with selected columns as $T$, the train-
The faithfulness of these statements were further checked by a verifier. As a result, LoFT can generate a diverse set of faithful statements covering different table regions and reasoning operations. For each table in the LOGIC-NLG test set, we randomly sampled five candidate statements for evaluation.

The logic form synthesis pipeline was first applied to synthesize candidate logic forms that cover different table regions and logical operations. LoFT is applied to generate statements for each candidate logic form. Then a statement verifier is used to filter out those potentially unfaithful statements. As a result, LoFT can generate a diverse set of faithful statements covering different table regions and reasoning operations. For each table in the LOGIC-NLG test set, we randomly sampled five candidate statements for evaluation.

To generate a candidate logic form, the pipeline first sampled a logic form using a random function and corresponding function definitions. Each function category corresponded to one unique table reasoning operation. For example, max/min, greater/less) into smaller groups to obtain a more abstract template. Each function category was defined to ensure that the generated candidate logic forms follow a similar distribution as LoFT. To instantiate the sampled template, a bottom-up sampling strategy is adopted to fill in each placeholder of the template and finally generate the logic form.

Statement Generation & Verification Through the logic form synthesis pipeline, we obtained a large number of candidate logic forms. For each logic form, we used LoFT to generate the corresponding statement. Then LoFT is trained to generate the reference statement given the translated logic form and serialized table data.
responding statement. The candidate statements might still contain some factually incorrectness, thus we applied an NLI-based verifier to filter out those potentially unfaithful generations. Specifically, we used the TABFACT (Chen et al., 2020b) dataset to train a classifier, which adopts RoBERTa-base as the backbone. We fed each generated statement and its corresponding table into the classifier, and only kept those statements that were predicted as entailed. Then we randomly sampled five statements as the output for each table in LOGiCNLG.

3.4 Enhancing LT2T via Logic Form Control
This subsection provides two perspectives to explain why logic forms can help improve both faithfulness and diversity of LT2T generation.

Logic Form as Content Planner Logic forms pass column or cell values as arguments, guiding the model to focus on relevant table regions. The function category of the logic form, such as count, helps the model better organize logical-level content planning.

Logic Form as Fact Verifier Logic forms are defined with unambiguous semantics, hence are reliable mediators to achieve faithful and controllable logical generations. During the inference stage, we synthesize candidate logic forms with 100% execution correctness. The sampled logic form serves as a fact verifier and conveys accurate logical-level facts for controllable LT2T generation.

4 Experimental Setup
We next discuss the evaluation metrics, baselines, and implementation details for the experiments.

4.1 Evaluation Metrics
We applied various automated evaluation metrics at different levels to evaluate the model performance from multiple perspectives.

Surface-level Following Chen et al. (2020a), we used BLEU-1/2/3 to measure the consistency of generated statements with the reference.

Diversity-level We used Distinct-n (Li et al., 2016) and self-BLEU-n (Zhu et al., 2018) to measure the diversity of five generated statements for each table. Distinct-n is defined as the total number of distinct n-grams divided by the total number of tokens in the five generated statements; Self-BLEU-n measures the average n-gram BLEU score between generated statements. We measured Distinct-2 and Self-BLEU-4 in our experiment.

Faithfulness-level Similar as the previous works (Chen et al., 2020a; Nan et al., 2022; Liu et al., 2022a), we used a parsing-based evaluation metric (i.e., SP-Acc) and two NLI-based evaluation metrics (i.e., NLI-Acc and TAPEX-Acc) to measure the faithfulness of generation. SP-Acc directly extracts the meaning representation from the generated sentence and executes it against the table to verify the correctness. NLI-Acc and TAPEX-Acc use TableBERT (Chen et al., 2020b) and TAPEX (Liu et al., 2022b) respectively as their backbones, and were finetuned on the TABFACT dataset (Chen et al., 2020b). Liu et al. (2022a) found that NLI-Acc is overly positive about the predictions, while TAPEX-Acc is more reliable to evaluate the faithfulness of generated sentences.

4.2 Baseline Systems
We implemented following baseline systems for the performance comparison: GPT2-TabGen (Chen et al., 2020a) directly fine-tunes GPT-2 over the LOGiCNLG dataset; GPT2-C2F (Chen et al., 2020a) first produces a template which determines the global logical structure, and then generates the statement conditioned on the template; DCSVED (Chen et al., 2021) applies a de-confounded variational encoder-decoder to reduce the spurious correlations during LT2T generation training; DEVTC (Perlitz et al., 2022) utilized reasoning operation types as an explicit control to increase the diversity of LT2T generation; and R2D2 (Nan et al., 2022) trains a generative language model both as a generator and a faithfulness discriminator with additional replacement detection and unlikelihood learning tasks, to enhance the faithfulness of LT2T generation.

4.3 Implementation Details
Following Shu et al. (2021), we converted each logic form into a more human-readable form for both LoFT training and inference data. LoFT was implemented using fairseq library (Ott et al., 2019), with BART-Large (Lewis et al., 2020) as the backbones. All experiments were conducted on an 8 NVIDIA RTX-A5000 24GB cluster. Both LoFT and the statement verifier was trained for 5,000 steps with a batch size of 128. The best checkpoints were selected by the validation loss.
5 Experimental Results

This section discusses automated and human evaluation results of different systems.

5.1 Main Results

Table 1 presents the results on LOGICNLG. LoFT outperforms all the baselines on the criteria of diversity and faithfulness, and is the first model that achieves state-of-the-art results on both faithfulness- and diversity-level. It is worth noting that in the LOGICNLG setting, a generated statement is allowed to cover a different table region or reasoning operations from the references, as long as it is fluent and factually correct. However, in such cases, the reference-based metrics will be low, explaining why the BLEU-1/2/3 scores of LoFT are lower than other models.

### 5.2 Human Evaluation

We conducted the human evaluation with four expert annotators using the following three criteria: (1) **Faithfulness** (scoring 0 or 1): if all facts contained in the generated statement are entailed by the table content; (2) **Diversity** (voting the best & worst): if the five generated statements cover information from different table regions, and use different reasoning operations; (3) **Fluency** (scoring 0 or 1): if the five generated statements are fluent and without any grammar mistakes.

We chose R2D2 (Nan et al., 2022) and DEVTC (Perlitz et al., 2022) for comparison, as they achieved best-performance results in faithfulness and diversity, respectively. We sampled 50 tables from the LOGICNLG test set. For each table, we selected all five generated statements from each model’s output. To ensure fairness, the model names were hidden to the annotators, and the display order between three models was randomly shuffled. Human evaluation results show that LoFT delivers improvements in both faithfulness (Table 3) and diversity (Table 2), while achieving comparable performance in fluency (Table 3).

### 6 Conclusions

This work proposes LoFT, which utilizes logic forms as fact verifiers and content planners to enable controllable LT2T generation. Experimental results on LOGICNLG demonstrate that LoFT delivers a great improvement in both diversity and faithfulness of LT2T generation.

**Limitations**

The first limitation of our approach is that LoFT does not explore long text generation (Moosavi et al., 2021). LoFT only supports the generation of multiple single sentences. To enable long text generation (i.e., generate a long paragraph that delivers...
various perspectives on the table data), a global content planner (Su et al., 2021) needs to be designed to highlight which candidate sentences should be mentioned and in which order. Additionally, we believe that LOFT can also be applied to text generation over hybrid context with both textual and tabular data (Chen et al., 2020c; Zhao et al., 2022a; Nakamura et al., 2022).

The second limitation of our work is that the statement verifier discussed in Section 3.3 was trained using the same data as NLI-Acc and TAPEX-Acc. This might bring some bias for NLI-based metrics on faithfulness-level evaluation. In the future, we will exploit a more robust automated evaluation system (Fabbri et al., 2021; Liu et al., 2022c) to comprehensively evaluate the LT2T model performances from different perspectives.

Moreover, we applied the SASP model (Ou and Liu, 2022) to convert statements into logic forms (Section 3.2). Some converted logic forms may be inconsistent with the original statement. We believe that future work could incorporate the Logic2Text (Chen et al., 2020d) dataset into training data to further improve the LOFT performance.

Ethical Consideration

We used the LOGICNLG (Chen et al., 2020a) dataset for training and inference. LOGICNLG is publicly available under MIT license\(^1\) and widely used in NLP research and industry.

References


Ao Liu, Haoyu Dong, Naoaki Okazaki, Shi Han, and Dongmei Zhang. 2022a. PLOG: Table-to-logic pretraining for logical table-to-text generation. In EMNLP 2022.


Yixin Liu, Alexander R. Fabbri, Pengfei Liu, Yilun Zhao, Linyong Nan, Ruilin Han, Simeng Han, Shaqiqi Joty, Chien-Sheng Wu, Caiming Xiong, and Dragomir Radev. 2022c. Revisiting the gold standard: Grounding summarization evaluation with robust human evaluation.

\(^1\)https://opensource.org/licenses/MIT


A Appendix
<table>
<thead>
<tr>
<th>Reasoning Op</th>
<th>Function Category</th>
<th>Name</th>
<th>Arguments</th>
<th>Output</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique</td>
<td>UNIQUE</td>
<td>only</td>
<td>view</td>
<td>bool</td>
<td>returns whether there is exactly one row in the view</td>
</tr>
<tr>
<td>Aggregation</td>
<td>AGGREGATION</td>
<td>avg/sum</td>
<td>view, header, string</td>
<td>number</td>
<td>returns the average/sum of the values under the header column</td>
</tr>
<tr>
<td>Count</td>
<td>COUNT</td>
<td>count</td>
<td>view</td>
<td>number</td>
<td>returns the number of rows in the view</td>
</tr>
<tr>
<td>Ordinal</td>
<td>ORD_ARG</td>
<td>nth_argmax/nth_argmin</td>
<td>view, header, string</td>
<td>view</td>
<td>returns the row with the n-th maximum value in header column</td>
</tr>
<tr>
<td></td>
<td>ORIGINAL</td>
<td>nth_max/nth_min</td>
<td>view, header, string</td>
<td>number</td>
<td>returns the n-th max/n-th min of the values under the header column</td>
</tr>
<tr>
<td></td>
<td>SUPER_ARG</td>
<td>argmax/argmin</td>
<td>view, header, string</td>
<td>view</td>
<td>returns the row with the maximum value in header column</td>
</tr>
<tr>
<td>Comparative</td>
<td>COMPARE</td>
<td>eq/not_eq</td>
<td>object, object</td>
<td>bool</td>
<td>returns if the two arguments are equal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>round_eq</td>
<td>object, object</td>
<td>bool</td>
<td>returns if the two arguments are roughly equal under certain tolerance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>greater/less</td>
<td>object, object</td>
<td>bool</td>
<td>returns if 1st argument is greater/less than 2nd argument</td>
</tr>
<tr>
<td></td>
<td></td>
<td>diff</td>
<td>object, object, object</td>
<td>object</td>
<td>returns the difference between two arguments</td>
</tr>
<tr>
<td>Majority</td>
<td>MAJORITY</td>
<td>all_eq/not_eq</td>
<td>view, header, string, object</td>
<td>bool</td>
<td>returns whether all the values under the header column are equal/not equal to 3rd argument</td>
</tr>
<tr>
<td></td>
<td></td>
<td>all_greater/less</td>
<td>view, header, string, object</td>
<td>bool</td>
<td>returns whether all the values under the header column are greater/less than 3rd argument</td>
</tr>
<tr>
<td></td>
<td></td>
<td>all_greater_eq/less_eq</td>
<td>view, header, string, object</td>
<td>bool</td>
<td>returns whether all the values under the header column are greater/less or equal to 3rd argument</td>
</tr>
<tr>
<td></td>
<td></td>
<td>most_eq/not_eq</td>
<td>view, header, string, object</td>
<td>bool</td>
<td>returns whether most of the values under the header column are equal/not equal to 3rd argument</td>
</tr>
<tr>
<td></td>
<td></td>
<td>most_greater/less</td>
<td>view, header, string, object</td>
<td>bool</td>
<td>returns whether most of the values under the header column are greater/less than 3rd argument</td>
</tr>
<tr>
<td></td>
<td></td>
<td>most_greater_eq/less_eq</td>
<td>view, header, string, object</td>
<td>bool</td>
<td>returns whether most of the values under the header column are greater/less or equal to 3rd argument</td>
</tr>
<tr>
<td>Conjunction</td>
<td>FILTER</td>
<td>filter_eq/not_eq</td>
<td>view, header, string, object</td>
<td>view</td>
<td>returns the subview whose values under the header column are equal/not equal to 3rd argument</td>
</tr>
<tr>
<td></td>
<td></td>
<td>filter_greater/less</td>
<td>view, header, string, object</td>
<td>view</td>
<td>returns the subview whose values under the header column are greater/less than 3rd argument</td>
</tr>
<tr>
<td></td>
<td></td>
<td>filter_greater_eq/less_eq</td>
<td>view, header, string, object</td>
<td>view</td>
<td>returns the subview whose values under the header column are greater/less or equal to 3rd argument</td>
</tr>
<tr>
<td>Other</td>
<td>OTHER</td>
<td>filter_all</td>
<td>view, header, string</td>
<td>view</td>
<td>returns the view itself for the case of describing the whole table</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hop</td>
<td>view, header string</td>
<td>object</td>
<td>returns the value under the header column of the row</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and</td>
<td>bool, bool</td>
<td>bool</td>
<td>returns the boolean operation result of two arguments</td>
</tr>
</tbody>
</table>

Table 4: A complete list of function definitions for the logic forms (Similar as Chen et al. (2020d)).