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

Yilun Zhao[*1] **Zhenting Qi**[*2] **Linyong Nan**[1]
Lorenzo Jaime Yu Flores[1] **Dragomir Radev**[1]
[1]Yale University [2] Zhejiang 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

2008 Champions Tour

Rank	Player	Country	Earnings	Wins
1	Hale Irwin	United States	24,920,665	45
2	Gil Morgan	United States	18,964,040	25
3	Dana Quigley	United States	14,406,269	11
4	Bruce Fleisher	United States	13,990,356	18
5	Larry Nelson	United States	13,262,808	19

Five statements generated by R2D2

Hale Irwin and Gil Morgan represent the same country
Hale Irwin had more wins than Gil Morgan
Hale Irwin had more earnings than Gil Morgan
Hale Irwin had the more wins than Bruce Fleisher
Bruce Fleisher had the highest earnings of any player with 13,990,356

Five statements generated by LoFT

The average earnings of Hale Irwin and Dana Quigley is 19,663,467
Five of the players are from the same country, United States
Dana Quigley had less wins than Larry Nelson
Most of the players had earnings less than 18,964,040
Larry Nelson had the least number of earnings

Figure 1: An example of logical table-to-text generation. (a) Statements generated by previous models (Nan et al., 2022): the generation suffers from 1) *Lack of diversity*, as three of the generated statements are focused on the same table regions (i.e., "Hale Irwin" and "Gil Morgan"), and three of them use the similar reasoning operations (i.e., comparative); 2) *Unfaithfulness*, as one of the generated statements is factually incorrect given the table content. (b) Statements generated by LoFT: By utilizing logic forms to *control* the generation, our method can generate multiple factually correct sentences that each use a different reasoning operation to offer various perspectives on the tabular data.

[*]Equal Contributions.

Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pages 554–561

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, LOGICNLG 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 LOGICNLG (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 unlikelihood 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

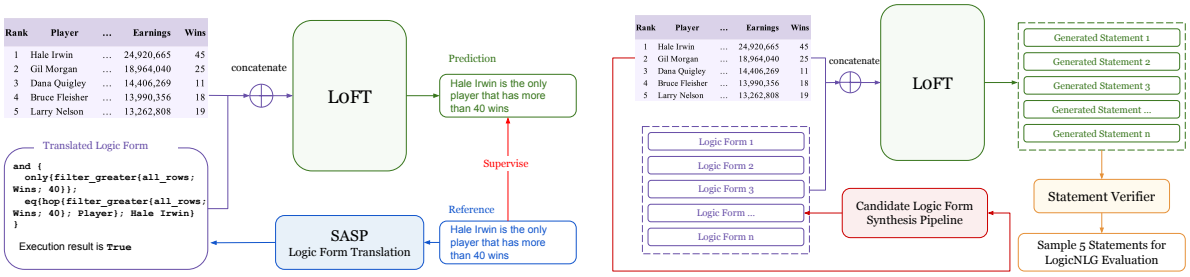
This section first introduces the logic form utilized, and then delves into the training and inference process of LOFT. We also explain how the use of logic forms can enhance both faithfulness and text-diversity in LT2T generation.

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-



(a) LoFT training stage.

(b) LoFT inference stage.

Figure 2: The illustration of LoFT. (a) During the training stage, the SASP model is first applied to translate each statement in the LogicNLG training set into the logic form. Then LoFT is trained to generate the reference statement given the translated logic form and serialized table data. (b) During the inference stage, given each table, 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 LogicNLG test set, we randomly sampled five candidate statements for evaluation.

ing objective of LoFT is to generate a sentence $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$ that is both fluent and faithful, with the translated logic form l as control.

$$\boldsymbol{y} = \text{argmax} \prod_{i=1}^{n} P(y_i | y_{<i}, T, l; \theta) \quad (1)$$

where θ denotes the parameters of a seq2seq LM.

Training Dataset Collection Since the LogicNLG dataset does not contain logic form annotations, we had to augment each statement in the training set with its corresponding logic forms. To construct *{statement, logic form}* parallel data for the LogicNLG training set, we adapted SASP (Ou and Liu, 2022), the state-of-the-art model for TabFact dataset, which leverages structure-aware semantic parsing over tables to translate the given statement into logic form. In this work, given an example in the LogicNLG training set, SASP was applied to generate its logic form, resulting in a total of 15,637 examples for LoFT training.

3.3 LoFT Inference

During the inference stage, for each given table, we first applied the logic form synthesis pipeline to synthesize multiple candidate logic forms (Liu et al., 2022a). For each of these logic forms, the system generates its corresponding statement. The faithfulness of these statements were further checked by a verifier.

Logic Form Synthesis Pipeline To synthesize a candidate set of logic forms paired with each supporting table, we applied a similar logic form synthesis pipeline as Liu et al. (2022a).

We extracted templates of logic forms from the collected LoFT training dataset. Specifically, we categorized functions with similar definitions (e.g., `max/min`, `greater/less`) into smaller groups to obtain a more abstract template. Each function category corresponded to one unique table reasoning skill. For each template, we masked specific entities in the logic forms as typed placeholders (i.e., `col` to denote a column header, `obj` to denote an object). Finally, we obtained 45 different templates, covering 8 table logical operations. Table 4 shows the complete list of reasoning operations and corresponding function definitions.

Given the table and each set of selected columns, the pipeline would synthesize a total of 20 candidate logic forms whose execution result over the table is `True`. To generate a candidate logic form, the pipeline first sampled a logic form using a weighted-sampling technique with the weight equal to the template distribution in the LoFT training dataset (Section 3.2). The weighted sampling is to ensure that the generated candidate logic forms follow a similar distribution as LogicNLG. 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 cor-

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; **DCVED** (Chen et al., 2021) applies a deconfounded 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.

Model	Surface-level	Diversity-level		Faithfulness-level		
	BLEU-1/2/3↑	Distinct-2↑	s-BLUE-4↓	SP-Acc↑	NLI-Acc↑	TAPEX-Acc↑
GPT2-TabGen (Chen et al., 2020a)	48.8/27.1/12.6	59.0	55.3	42.1	68.7	45.0
GPT2-C2F (Chen et al., 2020a)	46.6/26.8/13.3	60.3	52.8	42.7	72.2	44.1
DCVED* (Chen et al., 2021)	49.5/28.6/15.3	–	–	43.9	76.9	–
DEVTC‡ (Perlitz et al., 2022)	51.3/30.6/16.3	73.7	21.3	44.3	77.9	55.6
R2D2 (Nan et al., 2022)	51.8/32.4/18.6	60.1	51.5	50.8	85.6	60.2
LoFT	48.1/27.7/14.9	**79.5**	**17.7**	**57.7**	**86.9**	**61.8**

Table 1: Performance on the LogicNLG test set. ‡: results from our own implementation; ∗: code not released and we used the results reported in original papers. LoFT achieves great improvement on faithfulness and diversity.

Diversity Criteria	DEVTC		R2D2		LoFT	
	Best↑	Worst↓	Best↑	Worst↓	Best↑	Worst↓
Table Coverage	8	16	5	20	29	5
Reasoning Op	19	1	2	37	24	2

Table 2: Number of times the system was selected as best or worst by majority vote (including ties). LoFT outperforms other baselines in terms of diversity for both table coverage and reasoning operations.

Model	Faithfulness ↑ Agreement / κ	Fluency ↑ Agreement / κ
DEVTC	63.5 / 0.69	86.5 / 0.80
R2D2	71.5 / 0.73	**90.0** / 0.84
LoFT	**75.0** / 0.76	88.0 / 0.81

Table 3: Human evaluation results on the criteria of faithfulness and fluency, with the total agreement by Fleiss' Kappa (κ) (Fleiss, 1971). LoFT has the best performance in terms of faithfulness, while achieving comparable performance in fluency.

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 faithulness-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

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

Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, Shiyang Li, Xiyou Zhou, and William Yang Wang. 2020b. 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 Wang. 2020c. Hybridqa: A dataset of multi-hop question answering over tabular and textual data. *Findings of EMNLP 2020*.

Wenqing Chen, Jidong Tian, Yitian Li, Hao He, and Yaohui Jin. 2021. De-confounded variational encoder-decoder for logical table-to-text generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5532–5542, Online. Association for Computational Linguistics.

Zhiyu Chen, Wenhu Chen, Hanwen Zha, Xiyou Zhou, Yunkai Zhang, Sairam Sundaresan, and William Yang Wang. 2020d. Logic2text: High-fidelity natural language generation from logical forms.

Alexander R. Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2021. SummEval: Re-evaluating summarization evaluation. *Transactions of the Association for Computational Linguistics*, 9:391–409.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

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, Austin, Texas. Association for Computational Linguistics.

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, Online. Association for Computational Linguistics.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *NAACL 2016*.

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*.

Ao Liu, Congjian Luo, and Naoaki Okazaki. 2021. Improving logical-level natural language generation with topic-conditioned data augmentation and logical form generation. *arXiv preprint arXiv:2112.06240*.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2022b. TAPEX: Table pre-training via learning a neural SQL executor. In *International Conference on Learning Representations*.

Yixin Liu, Alexander R. Fabbri, Pengfei Liu, Yilun Zhao, Linyong Nan, Ruilin Han, Simeng Han, Shafiq 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

Nafise Sadat Moosavi, Andreas Rücklé, Dan Roth, and Iryna Gurevych. 2021. Scigen: a dataset for reasoning-aware text generation from scientific tables. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Kai Nakamura, Sharon Levy, Yi-Lin Tuan, Wenhu Chen, and William Yang Wang. 2022. HybriDialogue: An information-seeking dialogue dataset grounded on tabular and textual data. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 481–492, Dublin, Ireland. Association for Computational Linguistics.

Linyong Nan, Lorenzo Jaime Flores, Yilun Zhao, Yixin Liu, Luke Benson, Weijin Zou, and Dragomir Radev. 2022. R2D2: Robust data-to-text with replacement detection. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6903–6917, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Suixin Ou and Yongmei Liu. 2022. Learning to generate programs for table fact verification via structure-aware semantic parsing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7624–7638, Dublin, Ireland. Association for Computational Linguistics.

Ankur 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 the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online. Association for Computational Linguistics.

Yotam Perlitz, Liat Ein-Dot, Dafna Sheinwald, Noam Slonim, and Michal Shmueli-Scheuer. 2022. Diversity enhanced table-to-text generation via type control. *arXiv preprint arXiv:2205.10938*.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with entity modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2023–2035, Florence, Italy. Association for Computational Linguistics.

Chang Shu, Yusen Zhang, Xiangyu Dong, Peng Shi, Tao Yu, and Rui Zhang. 2021. Logic-consistency text generation from semantic parses. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4414–4426, Online. Association for Computational Linguistics.

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, Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

Xiaoyu Yang, Feng Nie, Yufei Feng, Quan Liu, Zhigang Chen, and Xiaodan Zhu. 2020. Program enhanced fact verification with verbalization and graph attention network. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7810–7825, Online. Association for Computational Linguistics.

Xiaoyu Yang and Xiaodan Zhu. 2021. Exploring decomposition for table-based fact verification. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1045–1052, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. 2022a. MultiHiertt: Numerical reasoning over multi hierarchical tabular and textual data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6588–6600, Dublin, Ireland. Association for Computational Linguistics.

Yilun Zhao, Linyong Nan, Zhenting Qi, Rui Zhang, and Dragomir Radev. 2022b. ReasTAP: Injecting table reasoning skills during pre-training via synthetic reasoning examples. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9006–9018, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research; Development in Information Retrieval*, SIGIR '18, page 1097–1100, New York, NY, USA. Association for Computing Machinery.

A Appendix

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

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