LogicToP: Logic Tree-of-Program with Table Instruction-tuned LLMs for Controlled Logical Table-to-Text Generation

Anonymous ACL submission

Abstract

Logical table-to-text generation aims to generate natural language descriptions that fluently and precisely describe the given table with both surface-level and logic-level fidelity. Although large language models (LLMs) have demonstrated strong capabilities in plain text, their proficiency in interpreting and reasoning tabular data is still limited. In this paper, we are the first to comprehensively explore the performance of various LLMs in the logical table-to-text generation task. However, we find that existing LLMs are difficult to achieve satisfactory results in this task. Even worse, existing prompt strategies cannot cope with complex non-chain logical reasoning scenarios on tables. To address the challenges mentioned above, we constructed a new table-related instruction dataset called LogicTableInstruct and instruction-tuned two open-source LLMs on this dataset, resulting in two specialized LLMs for table-related tasks. We also introduced a novel reasoning framework termed Logic Tree-of-Program (LogicToP) to improve the logical reasoning ability of the LLMs on tables. Our extensive experiments on various LLMs demonstrated that LogicToP can effectively improve the performance of LLMs on this task. Our LogicTableLLaMA-3.1-8B model in the 5-shot LogicToP setting achieves state-of-the-art results on the Logic2Text dataset. The code and data will be released to boost future work on table-related tasks.

1 Introduction

Logical table-to-text generation (LT2T) is an important branch of Natural Language Generation (NLG). It aims to generate natural language descriptions that fluently and precisely describe the given table with both surface-level and logic-level fidelity. Early methods (Liu et al., 2022; Deng et al., 2023; Zhao et al., 2023b) mainly followed the paradigm of "pre-training and fine-tuning" to develop customized pre-training strategies for LT2T tasks based on Pre-trained Language Models (PLMs) such as GPT-2 (Radford et al., 2019), BART (Lewis et al., 2019), and T5 (Raffel et al., 2020). However, it is difficult to generalize and requires a significant amount of resources to pre-train and fine-tune for every new task.

In recent years, Large Language Models (LLMs) have showcased notable proficiency in handling various tasks in NLP through In-Context Learning (ICL), which incorporates input-output demonstrations into the prompt. Compared with specific table pre-training, using table-related instruction data to fine-tune general LLMs (Li et al., 2023; Zhang et al., 2024; Zhuang et al., 2024) has become a more efficient paradigm called "table instruction-tuning", which saves a lot of resources and time for pre-training and can generalize to unseen tasks. To further enhance the reasoning ability of LLMs, various prompting strategies such as CoS (Hu et al., 2023), SymbCoT (Xu et al., 2024), and PoT (Chen et al., a) originating from Chain-of-Thoughts prompting (Wei et al., 2022) have been proposed. However, existing prompt strategies cannot cope with complex non-chain logical reasoning scenarios on tables. In addition, there is currently no in-depth research on the performance of table instruction-tuned LLMs in the LT2T task.

In this paper, we first comprehensively explored the performance of various LLMs in the LT2T task. Unfortunately, we found that existing LLMs are difficult to achieve satisfactory results. Thus, we constructed a new table-related instruction dataset called LogicTableInstruct and instruction-tuned two open-source general LLMs on the LogicTableInstruct, resulting in two specialized LLMs for table-related tasks. Moreover, we introduced a novel reasoning framework termed LogicToP for controlled logical table-to-text generation to solve the problem of existing prompt strategies being unable to effectively perform logical reasoning in intricate non-chain structure scenarios.

In summary, our main contributions are:

- We are the first to comprehensively explore the performance of different LLMs in the controlled logical table-to-text generation task.
- We constructed a new table-related instruction dataset called LogicTableInstruct, which includes 9 different table-related tasks.
- We developed two instruction-tuned LLMs (*LogicTableLLaMA-3.1-8B* and *Qwen2.5-Table-7B-Instruct*) specifically designed for table-related tasks.
- We introduced a novel reasoning framework termed LogicToP for the controlled logical table-to-text generation task. Our extensive experiments on various LLMs demonstrated that LogicToP can effectively improve performance on this task. Our *LogicTableLLaMA-3.1-8B* model in the 5-shot LogicToP setting achieves state-of-the-art results on the Logic2Text dataset.

2 Related Work

2.1 Logical Table-to-Text Generation

Logical table-to-text generation (LT2T) needs both surface-level fidelity (i.e., demanding that the generated text accurately represents the underlying data) and logic-level fidelity (i.e., going beyond superficial facts to ensure that the generated text is logically entailed by the given table). In the era of the "pre-train and fine-tune" paradigm, most logical table-to-text generation systems adopt pre-trained language models (PLMs, such as GPT-2 (Radford et al., 2019), BART (Lewis et al., 2019), and T5 (Raffel et al., 2020)), either by using existing pre-trained language models (Chen et al., 2020a, 2021; Shi et al., 2022; Nan et al., 2022; Perlitz et al., 2022; Zhao et al., 2023a; Alonso and Agirre, 2024; Wu and Hou, 2024) or by developing pre-training strategies tailored for the table-to-text generation task (Liu et al., 2022; Deng et al., 2023; Zhao et al., 2023b). After entering the large language models (LLMs) era, researchers (Li et al., 2023; Zhang et al., 2024; Zhuang et al., 2024) adopted a new paradigm called "table instruction-tuning". This paradigm uses table-related instruction data to fine-tune the general LLMs and reduce the huge cost of the pre-training process. However, there is currently no in-depth exploration of the performance of instruction-tuned large language models in the logical table-to-text generation task.

2.2 Chain-of-Thoughts Reasoning with LLMs

Although LLMs have demonstrated remarkable efficacy in a variety of NLP tasks, their reasoning ability is frequently viewed as a disadvantage. Even worse, this ability cannot be obtained by merely expanding the model's size. When given the Chain-of-Thoughts prompting (Wei et al., 2022), LLMs have lately been discovered capable of complex reasoning over text. In addition, the Chain-of-Symbol (CoS; Hu et al., 2023) uses condensed symbolic chain representations to depict complicated contexts during symbolic reasoning planning. Symbolic Chain-of-Thought (SymbCoT; Xu et al., 2024) integrates symbolic expressions and logic rules with CoT prompting to strengthen the logical reasoning capability of LLMs. By introducing programming languages to describe the reasoning process, the Program-of-Thoughts (PoT; Chen et al., a) and PAL (Gao et al., 2023) convert the reasoning problem into an executable program to derive the answer. To solve the problem that the original chain structure naturally limits the scope of exploration, Tree-of-Thoughts (ToT; Yao et al., 2023), a variant of CoT, allows LLMs to perform deliberate decision-making by considering multiple different reasoning paths and self-evaluating choices to decide the next course of action. Skeleton-of-Thought (SoT; Ning et al., 2023) is another variation of ToT that breaks down a problem into smaller, parallel-processable problems. Different from the above studies, we focus on using the Logic Tree-of-Program to enhance the logical reasoning ability of LLMs in logical table-to-text generation.

3 Methodology

3.1 Problem Definition

Given a table T with its title C, the task of controlled logical table-to-text generation (CLT2T) is to generate a description $Y = (y_1, y_2, \cdots, y_{|Y|})$ that is both fluent and logically consistent, with the logical type L as control (Perlitz et al., 2022). $T = \{T_{i,j} | 1 \leq i \leq N_{row}, 1 \leq j \leq N_{col}\}$, where the N_{row} and N_{col} are the numbers of rows and columns, respectively, and $T_{i,j}$ is the table cell value at row i and column j. Unlike previous methods (Chen et al., 2020b; Zhao et al., 2023b), we did not directly provide the standard logical form corresponding to Y in the input, as we believe their

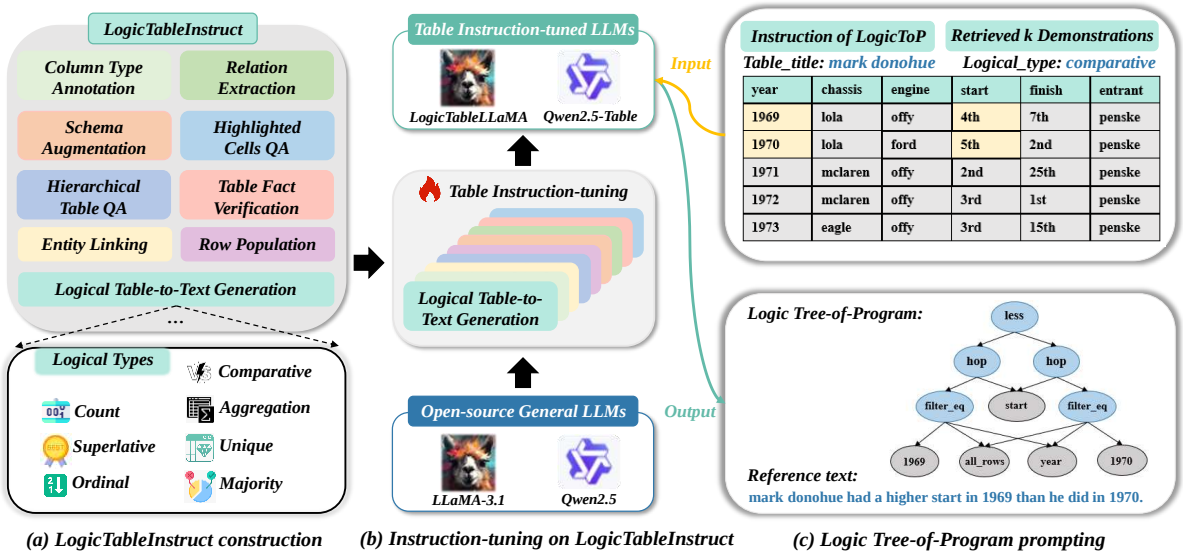

Figure 1: Overview of our LogicToP framework.

methods were leaking the answer. Following previous works (Parikh et al., 2020; Liu et al., 2022) towards controlled table-to-text generation, we also incorporate highlighted cells (or columns) H in the input as additional supervision signals. The task objective thus becomes $P(Y|T, C, L, H)$, as shown in the Equ(1):

$$P(Y|T, C, L, H) = \prod_{i=1}^{|Y|} p_{LM}(y_i|T, C, L, H, y_{<i}) \quad (1)$$

where p_{LM} is a probabilistic language model.

3.2 The LogicToP Framework

We elaborate on the LogicToP framework in the following three steps, as shown in the Figure 1.

(1) *LogicTableInstruct* construction

Although general LLMs can generate fluent text, they lack sufficient logical reasoning ability to generate logically faithful table descriptions. Using table-related instruction data to fine-tune a general LLM can enhance its performance in table-related tasks. However, existing works (Li et al., 2023; Zhang et al., 2024) have not explored the logical table-to-text generation task, resulting in a lack of instruction data related to this task. Therefore, we constructed a new table-related instruction dataset called *LogicTableInstruct* based on the TableInstruct (Zhang et al., 2024). We retained the diversity of task categories in the original TableInstruct dataset but adjusted the ratio between the original data and the added data related to logical tables to 1:1. Specifically, as shown in Table 1, the training data corresponding to the logical table-to-text generation task contains a total of 36.8K samples, while the other eight in-domain tasks each contain 4.6K randomly selected samples. This data ratio is conducive to ensuring that the fine-tuned model has logical reasoning ability for tables, as well as the ability to understand, answer questions, and verify facts about tables. As shown on the left in the Figure 1, the instruction data for the logical table-to-text generation task covers seven logical types: count, superlative, ordinal, comparative, aggregation, unique, and majority. Please refer to Figure 5 for the definitions of logical types and examples of their logic forms. The visualization of quantity statistics for different logical types in the LogicTableInstruct dataset is shown in Figure 6.

(2) Instruction-tuning on *LogicTableInstruct*

The instructions let LLMs quickly adapt to a specific domain by constraining the model's outputs to match the intended response characteristics or domain knowledge without requiring extensive retraining or architecture design (Zhang et al., 2023). Considering that the 70B-level LLMs (LLaMA-3.1-70B-Instruct and Qwen2.5-72B-Instruct) perform better in open-source general LLMs under few-shot settings, we chose their same series but smaller-scale LLMs (LLaMA-3.1-8B-Instruct and Qwen2.5-7B-Instruct) as the backbone models for table instruction-tuning with our *LogicTableInstruct* dataset, ultimately resulting in two table instruction-tuned LLMs: *LogicTableLLaMA-3.1-8B* and *Qwen2.5-Table-7B-Instruct*, as shown in the middle of the Figure 1.

Task Category	Task Name	Original Dataset	In-domain	# Train	# Test
Table Interpretation	Column Type Annotation	TURL	YES	4.6K	0.5K
	Relation Extraction		YES	4.6K	0.5K
	Entity Linking		YES	4.6K	0.5K
Table Augmentation	Schema Augmentation	TURL	YES	4.6K	0.5K
	Row Population		YES	4.6K	0.3K
Question Answering	Hierarchical Table QA	HiTab	YES	4.6K	0.5K
	Highlighted Cells QA	FeTaQA	YES	4.6K	0.8K
Fact Verification	Fact Verification	TabFact	YES	4.6K	0.5K
Table-to-Text Generation	Logical Table-to-Text Generation	CONTLOG	YES	8.6K	1.1K
		LOFT	YES	28.2K	3.0K

Table 1: LogicTableInstruct dataset statistics grouped by task category.

(3) Logic Tree-of-Program prompting

To address the issue of CoS, SymbCoT, and PoT being unable to perform logical reasoning in complex non-chain structure scenarios, we propose a new reasoning method called the Logic Tree-of-Program (**LogicToP**) for controlled logical table-to-text generation. As shown in Figure 1 (c), before generating descriptive text, LogicToP requires the model to iteratively reason logically to form a logic tree with functions or cell contents as nodes. Please refer to the complete list of the function definitions and descriptions in Appendix E. In addition, in order to enable the model to generate more accurate logical forms, we dynamically retrieve multiple demonstration examples with the same logical type as the test sample for In-Context Learning during inference. Under the guidance of the logic tree, the text generated by the model unequivocally demonstrates superior fidelity, both at the surface level and in logical coherence.

4 Experimental Results

4.1 Experimental Settings

Here, we introduce the datasets, automatic evaluation metrics, and baselines used in our experiments.

4.1.1 Datasets

There are two benchmark datasets for logical table-to-text generation: **Logic2Text** (Chen et al., 2020b) and **LogicNLG** (Chen et al., 2020a). The samples in both datasets are open-domain tables scraped from Wikipedia. Each table is accompanied by several related sentences covering diverse types of logical inference. SASP (Ou and Liu, 2022) was used to construct the logic form of the examples from the LogicNLG training set, obtaining 15,637 examples in total for LOFT (Zhao et al., 2023b) training. Besides, Liu et al. (2022) re-organized the Logic2Text dataset by detecting highlighted cells based on their annotated logical forms, resulting in a new complementary dataset (CONTLOG) towards controlled logical table-to-text generation. In this work, we conduct experiments on two datasets, **CONTLOG** and **LOFT**, which have completed the logical form and highlighted cells based on tables.

4.1.2 Automatic Evaluation Metrics

We evaluate the generated description text from the following two aspects:

(1) We first assessed the informativeness of the generated texts using **BLEU** (Papineni et al., 2002) and **ROUGE** (Lin, 2004), which measure lexical similarity by calculating the overlap of n-gram at the word level between the generated texts and the ground-truth descriptions.

(2) Following previous works (Chen et al., 2020a), we also employ **SP-Acc** and **NLI-Acc** to evaluate the logical fidelity of the generated texts.

4.1.3 Baselines

In these experiments, we mainly take into account the following baseline methods based on PLMs and LLMs, respectively:

(1) PLMs-based Methods

GPT-TabGen (Chen et al., 2020a) directly fine-tunes GPT-2-medium (Radford et al., 2019) over Logic2Text and LogicNLG datasets. **DCVED** (Chen et al., 2021) is a de-confounded variational encoder-decoder based on causal intervention to mitigate spurious correlations in generations. **DEVTC** (Perlitz et al., 2022) takes a distinct strategy by using GPT-2-medium (Radford et al., 2019) in combination with reasoning operation types as explicit controls. **LogicMoE** (Wu and Hou, 2024) is a dedicated Mixture-of-Experts (MoE) model

Method	Surface-level Fidelity				Logical Fidelity	
	BLEU-1	BLEU-2	BLEU-3	ROUGE-L	SP-Acc	NLI-Acc
PLMs-based methods under fully-supervised setting						
GPT-TabGen (Chen et al., 2020a)	46.5	30.9	19.9	-	42.4	66.5
DCVED (Chen et al., 2021)	46.4	31.2	20.1	-	43.7	71.9
DEVTC (Perlitz et al., 2022)	47.8	32.6	22.2	-	41.9	74.4
LogicMoE (Wu and Hou, 2024)	**51.4**	**36.8**	**26.4**	-	**48.9**	**75.9**
LLMs-based methods under few-shot setting						
TableLlama 5-shot ICL	04.9	02.3	01.4	09.4	20.2	57.3
TableLlama 5-shot LogicToP	04.1↓	01.9↓	01.1↓	08.5↓	21.1↑	61.0↑
GPT-4o mini 5-shot ICL	28.0	17.5	11.7	35.9	47.8	88.6
GPT-4o mini 5-shot LogicToP	35.1↑	23.0↑	16.0↑	41.5↑	**54.8**↑	89.3↑
GLM-4-9B-Chat 5-shot ICL	12.5	07.0	04.1	22.4	24.4	89.7
GLM-4-9B-Chat 5-shot LogicToP	22.8⇑	12.4↑	07.2↑	35.8⇑	48.5⇑	**94.6**↑
LLaMA-3.1-8B-Instruct 5-shot ICL	19.9	12.2	08.0	31.6	48.6	80.7
LLaMA-3.1-8B-Instruct 5-shot LogicToP	30.5⇑	20.4↑	14.2↑	41.3↑	44.5↓	79.4↓
LLaMA-3.1-70B-Instruct 5-shot ICL	26.6	16.2	10.6	38.6	47.9	85.7
LLaMA-3.1-70B-Instruct 5-shot LogicToP	42.8⇑	29.3⇑	21.4⇑	42.3↑	52.7↑	87.3↑
♣ LogicTableLLaMA-3.1-8B 5-shot ICL	49.3	35.5	26.9	49.8	47.1	84.1
♣ LogicTableLLaMA-3.1-8B 5-shot LogicToP	**56.5**↑	**43.5**↑	**34.4**↑	**55.5**↑	51.8↑	87.1↑
Qwen2.5 series LLMs-based methods under few-shot setting						
Qwen2.5-7B-Instruct 5-shot ICL	21.5	12.3	07.7	31.6	39.4	81.4
Qwen2.5-7B-Instruct 5-shot LogicToP	31.4↑	20.2↑	14.0↑	37.3↑	53.3⇑	89.5↑
Qwen2.5-72B-Instruct 5-shot ICL	25.9	16.3	11.1	35.8	41.8	90.4
Qwen2.5-72B-Instruct 5-shot LogicToP	31.2↑	19.9↑	13.8↑	**41.7**↑	**54.6**⇑	**94.3**↑
Qwen2.5-Coder-7B-Instruct 5-shot ICL	29.0	17.0	11.1	38.9	52.1	83.3
Qwen2.5-Coder-7B-Instruct 5-shot LogicToP	**41.6**⇑	**28.0**⇑	**20.2**↑	41.4↑	54.4↑	88.6↑
Qwen2.5-Math-7B-Instruct 5-shot ICL	02.6	01.4	00.8	04.5	09.1	15.5
Qwen2.5-Math-7B-Instruct 5-shot LogicToP	02.7↑	01.5↑	01.0↑	04.2↓	11.6↑	22.4↑
♣ Qwen2.5-Table-7B-Instruct 5-shot ICL	30.8	16.8	10.6	33.2	46.9	69.8
♣ Qwen2.5-Table-7B-Instruct 5-shot LogicToP	40.6↑	25.4↑	17.2↑	38.9↑	51.3↑	76.3↑

Table 2: Performance comparisons of the automatic evaluation on the Logic2Text dataset. ♣ denotes our table instruction-tuned LLMs. ↑ and ↓ indicate whether LogicToP has improved or decreased compared to ICL, and ⇑ indicates that LogicToP has improved by more than 10%.

tailored for the LT2T task. In addition, we have chosen three more powerful methods than GPT-TabGen and DCVED as baselines for the Logic-NLG dataset: **R2D2** (Nan et al., 2022) incorporates additional replacement detection and unlikelihood learning tasks to train the T5-base (Raffel et al., 2020) to act as both a generator and a fidelity discriminator. **PLOG** (Liu et al., 2022) is a pre-trained logical form generator model based on BART-large (Lewis et al., 2019) to achieve more faithful LT2T. **LoFT** (Zhao et al., 2023b) based on BART-large controls the creative process by using logic forms as content planners and fact validators. We directly use the results reported in the papers corresponding to the above methods for comparison.

(2) Large Language Models (LLMs)

In this paper, we also add a baseline method (In-Context Learning) that directly uses the following LLMs to accomplish the controlled logical table-to-text generation task in a few-shot manner.

GPT-4o mini (Achiam et al., 2023). It is the most cost-efficient closed-source general LLM of ChatGPT (or GPT-4). The GPT-4o mini outperforms other small models on reasoning tasks involving text and vision. It scored 82.0% on MMLU, which is a benchmark for textual intelligence and reasoning.

TableLlama (Zhang et al., 2024). It is the first open-source generalist LLM instruction-tuned on a constructed TableInstruct dataset using LongLoRA (Chen et al., b) based on Llama 2 (7B) (Touvron et al., 2023) as the backbone model.

GLM-4-9B-Chat (GLM et al., 2024). It demonstrates comparable performance to GPT-4 and Claude 3 Opus in Chinese math and logic reasoning capabilities, though it lags behind GPT-4 Turbo.

The family of LLaMA-3.1 models[1]**.** We choose the following advanced models for experimentation: LLaMA-3.1-8B-Instruct and LLaMA-3.1-70B-Instruct.

The family of Qwen2.5 (Yang et al., 2024) models. We chose the following advanced models for extensive experimentation: Qwen2.5-7B-Instruct, Qwen2.5-72B-Instruct, Qwen2.5-Coder-7B-Instruct, and Qwen2.5-Math-7B-Instruct.

[1]https://ai.meta.com/blog/meta-llama-3-1/

Method	Surface-level Fidelity				Logical Fidelity	
	BLEU-1	BLEU-2	BLEU-3	ROUGE-L	SP-Acc	NLI-Acc
PLMs-based methods under fully-supervised setting						
DEVTC (Perlitz et al., 2022)	50.8	29.2	15.2	-	45.6	77.0
R2D2 (Nan et al., 2022)	51.8	32.4	18.7	**36.8**	50.8	85.6
PLOG (Liu et al., 2022)	**54.9**	**35.0**	**21.0**	-	50.5	**88.9**
LOFT (Zhao et al., 2023b)	48.1	27.7	14.9	-	**57.7**	86.9
LogicMoE (Wu and Hou, 2024)	54.6	33.5	19.4	-	49.2	82.7
LLMs-based methods under few-shot setting						
TableLlama 5-shot ICL	01.1	00.4	00.3	03.6	09.2	19.6
TableLlama 5-shot LogicToP	01.4↑	00.6↑	00.4↑	04.0↑	10.6↑	25.6↑
GPT-4o mini 5-shot ICL	14.2	06.3	03.4	21.0	38.2	70.9
GPT-4o mini 5-shot LogicToP	16.0↑	07.4↑	04.3↑	22.8↑	**53.7**⇑	76.7↑
GLM-4-9B-Chat 5-shot ICL	09.0	03.8	02.0	14.6	24.4	77.8
GLM-4-9B-Chat 5-shot LogicToP	11.5↑	05.0↑	02.7↑	17.3↑	48.5⇑	**84.0**↑
LLaMA-3.1-8B-Instruct 5-shot ICL	10.3	04.5	02.6	14.0	41.8	65.1
LLaMA-3.1-8B-Instruct 5-shot LogicToP	12.2↑	05.6↑	03.3↑	19.4↑	45.2↑	66.3↑
LLaMA-3.1-70B-Instruct 5-shot ICL	14.1	06.4	03.7	18.2	40.1	63.2
LLaMA-3.1-70B-Instruct 5-shot LogicToP	19.2↑	09.4↑	05.5↑	25.5↑	48.8↑	68.4↑
♣ LogicTableLLaMA-3.1-8B 5-shot ICL	17.5	08.8	05.4	22.0	46.2	55.5
♣ LogicTableLLaMA-3.1-8B 5-shot LogicToP	**29.9**⇑	**17.9**↑	**11.8**↑	**34.8**⇑	51.0↑	65.6⇑
Qwen2.5 series LLMs-based methods under few-shot setting						
Qwen2.5-7B-Instruct 5-shot ICL	13.7	05.9	03.3	20.1	40.8	71.3
Qwen2.5-7B-Instruct 5-shot LogicToP	14.4↑	06.2↑	03.5↑	20.2↑	52.7⇑	77.5↑
Qwen2.5-72B-Instruct 5-shot ICL	13.7	06.2	03.4	20.8	34.7	76.5
Qwen2.5-72B-Instruct 5-shot LogicToP	15.5↑	07.2↑	04.2↑	22.7↑	46.8⇑	**82.8**↑
Qwen2.5-Coder-7B-Instruct 5-shot ICL	15.0	06.8	03.9	21.7	42.2	58.1
Qwen2.5-Coder-7B-Instruct 5-shot LogicToP	**18.7**↑	08.9↑	05.0↑	**27.0**↑	**52.8**⇑	70.7⇑
Qwen2.5-Math-7B-Instruct 5-shot ICL	06.4	01.8	01.2	08.3	03.3	61.9
Qwen2.5-Math-7B-Instruct 5-shot LogicToP	06.3↓	01.8	01.2	08.2↓	03.7↑	62.6↑
♣ Qwen2.5-Table-7B-Instruct 5-shot ICL	13.9	06.4	03.8	19.2	46.3	49.0
♣ Qwen2.5-Table-7B-Instruct 5-shot LogicToP	**18.7**↑	**09.0**↑	**05.3**↑	23.8↑	51.1↑	55.2↑

Table 3: Performance comparisons of the automatic evaluation on the LogicNLG dataset. ♣ denotes our table instruction-tuned LLMs. ↑ and ↓ indicate whether LogicToP has improved or decreased compared to ICL, and ⇑ indicates that LogicToP has improved by more than 10%.

4.1.4 Implementation Details

The GPT-4o mini has a context window of 128K tokens, supports up to 16K output tokens per request, and has knowledge up to October 2023. We use the same hyperparameters for LLMs: temperature = 0.001, penalty = 1.2, max_new_tokens = 1024. For the table instruction-tuning, we provide detailed parameter settings in Appendix A.

4.2 Main Results and Analysis

Table 2 and Table 3 present the comparison of automatic evaluation results between LogicToP and other baselines on the Logic2Text and LogicNLG datasets, respectively. Compared with methods based on PLMs that have undergone specialized pre-training of tables, directly using a general large language model with 5-shot In-Context Learning (ICL) demonstrations does not perform well in logical table-to-text generation tasks. It is disappointing that the TableLlama model, which has been fine-tuned with various table-related instruction data, performs extremely poorly on this task. However, our proposed LogicToP method achieves state-of-the-art performance on the Logic2Text dataset using our table instruction-tuned LLMs such as LogicTableLLaMA-3.1-8B. On the LogicNLG dataset, our method significantly improved the logical reasoning ability of LLMs on tables. Specifically, we further analyze the experimental results through the following three perspectives:

(1) LogicToP vs. ICL. As shown in Table 2 and Table 3, we use ↑ and ↓ to indicate whether LogicToP has improved or decreased compared to ICL. For LLMs with strong generalization ability, the LogicToP method enables them to think about obtaining the corresponding logic tree-of-program before generating text, resulting in a significant performance improvement. For the TableLlama and Qwen2.5-Math-7B-Instruct models, which had poor performance, LogicToP has little or even a negative effect on their improvement. The reason is that these models can not correctly understand and follow the instructions of LogicToP.

(2) Different sizes of LLMs. When the results of LLaMA-3.1-8B-Instruct and LLaMA-3.1-70B-Instruct in the 5-shot LogicToP setting are com-

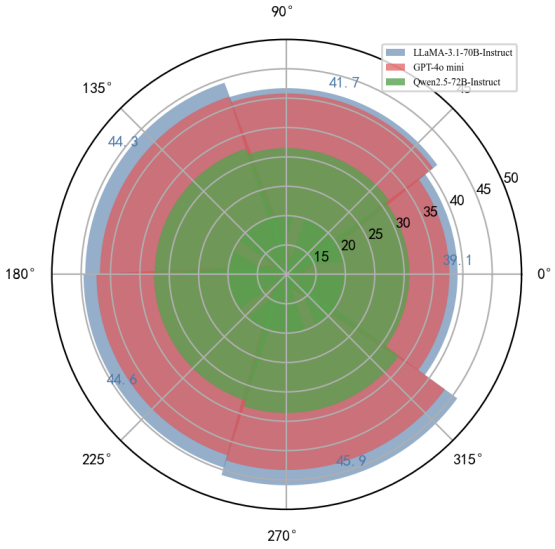

Figure 2: The results of different numbers of demonstrations (from 1-shot to 5-shot) on the Logic2Text dataset.

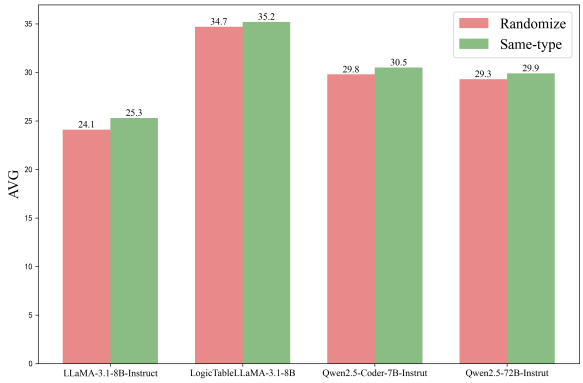

Figure 3: Comparison results of different selection strategies (Randomized vs. Same-type) for demonstration examples on the LogicNLG dataset.

pared, it is clear that LLaMA-3.1-70B-Instruct always outperforms LLaMA-3.1-8B-Instruct. Due to LLaMA-3.1-8B-Instruct's weak ability to follow instructions of LogicToP, its self-generated logic tree is prone to error, resulting in a decrease in its logical fidelity on the Logic2Text dataset. However, the gap between Qwen2.5-7B-Instruct and Qwen2.5-72B-Instruct is relatively small. Qwen2.5-72B-Instruct only has stable advantages in ROUGE-L and NLI-Acc metrics.

(3) General LLM vs. Special LLM. To explore whether using special domain data (such as code and math corpus) for continual pre-training (CPT) general LLMs can improve the performance on this logical table-to-text generation task, we chose the Qwen2.5 series LLMs for comparison. We can see from Table 2 and 3 that the overall performance of Qwen2.5-Coder-7B-Instruct is better than that of Qwen2.5-7B-Instruct, while the opposite is true for Qwen2.5-Math-7B-Instruct. Qwen2.5-Coder-7B-Instruct even surpassed Qwen2.5-72B-Instruct in BLEU-1/2/3 metrics, indicating that utilizing code data for CPT is helpful for this task.

4.3 Ablation Study

In the ablation experiment, we use the arithmetic mean (AVG) of six automatic evaluation metrics scores to represent overall performance. We further explore the factors that affect the performance of the LLMs through the following two aspects:

(1) Different number of demonstrations. To investigate the impact of the number of demonstration examples on the LLMs, we tested the performance of three LLMs (GPT-4o mini, LLaMA-3.1-70B-Instruct, and Qwen2.5-72B-Instruct) with a context window size of 128k from 1-shot to 5-shot LogicToP on the Logic2Text dataset. As shown in Figure 2, all three models exhibit a stable overall performance growth trend from 1-shot to 5-shot LogicToP. This leads to the conclusion that the large language model performs better on the task with more demonstration examples in the input. Under the same number of demonstration examples, the overall performance of LLaMA-3.1-70B-Instruct consistently outperforms GPT-4o mini and Qwen2.5-72B-Instruct.

(2) Randomized samples vs. Same-type samples. To explore the impact of different demonstration selection strategies on the LLMs, we tested the performance of four LLMs (LLaMA-3.1-8B-Instruct, LogicTableLLaMA-3.1-8B, Qwen2.5-Coder-7B-Instrut and Qwen2.5-72B-Instruct) using two strategies (Randomized samples and Same-type samples) in the 5-shot LogicToP setting on the LogicNLG dataset. As shown in Figure 3, the average score of the same-type sampling strategy is always slightly better than that of the randomized sampling strategy on these four LLMs. It also aligns with our intuition that using examples of the same logical type can help the model generate the logical form corresponding to the current test sample through similar logical forms. Moreover, LogicTableLLaMA-3.1-8B has improved its AVG score by about ten points compared to its backbone model, LLaMA-3.1-8B-Instruct, indicating that instruction-tuning with our LogicTableInstruct dataset can effectively enhance the model's logical reasoning ability on tables.

LLaMA-3.1-8B-Instruct 5-shot ICL

Here is the natural language description: The Capital Region of Denmark and the Region of Southern Denmark have the same English names, ""Capital Region of Denmark"" and ""Region of Southern Denmark"", respectively. However, the Capital Region of Denmark has a smaller population (1645825) compared to the Region of Southern Denmark (1194659), and also compared to the Region of Zealand (819427).

LLaMA-3.1-8B-Instruct 5-shot LogicToP

To solve this problem, I will follow these steps:
1. **Identify the logical type**: The logical type is ""comparative"".
2. **Determine the column and rows to compare**: The highlighted cells indicate that we need to compare the values in the ""population (January 1, 2008)"" column for rows 0 and 4.
3. **Compare the values**: The value for row 0 is 1645825, and the value for row 4 is 1194659.
4. **Generate the description**: Since the value for row 0 is greater than the value for row 4, the description will be: ""The capital region of denmark had a larger population than the region of southern denmark in 2008.""

LLaMA-3.1-70B-Instruct 5-shot LogicToP

logic_str: greater { row_0 ; population (january 1 , 2008) ; row_4 } = true
Answer: the capital region of denmark has a larger population than the region of southern denmark .

LogicTableLLaMA-3.1-8B 5-shot LogicToP

###The logical form is: greater { hop { filter_eq { all_rows ; english name ; capital region of denmark } ; population (january 1 , 2008) } ; hop { filter_eq { all_rows ; english name ; region of southern denmark } ; population (january 1 , 2008) } } = true
###Answer: the capital region of denmark has a higher population than the region of southern denmark

Reference Description and Logical_Form

Golden_logical_form: greater { hop { filter_eq { all_rows ; english name ; capital region of denmark } ; population (january 1 , 2008) } ; hop { filter_eq { all_rows ; english name ; region of southern denmark } ; population (january 1 , 2008) } } = true
Golden_description: the capital region of denmark has a higher population than the region of southern denmark.

Figure 4: Top 4 boxes: descriptions generated by different settings (5-shot ICL and 5-shot LogicToP) with various LLaMA-3.1 series LLMs. Bottom: reference description and logical form of $Test_{1089}$ on the Logic2Text dataset.

4.4 Case Study

To understand the effect of our LogicToP method more intuitively, we select one representative example ($Test_{1089}$) on the Logic2Text dataset and present its descriptions generated by different settings (5-shot ICL and 5-shot LogicToP) with various LLaMA-3.1 series LLMs in Figure 4. For the same model, LLaMA-3.1-8B-Instruct, there are two obvious errors in the description generated by the 5-shot ICL, while the 5-shot LogicToP generates the correct description. However, the instruction following and schema learning ability of the LLaMA-3.1-8B-Instruct model is weaker than that of the LLaMA-3.1-70B-Instruct model, which is reflected in the fact that the reasoning strategy of the LLaMA-3.1-8B-Instruct model remains at a step-by-step chain (**Identify the logical type*, *Determine the column and rows to compare*, *Compare the values*, and *Generate the description**) without generating the logical form in the demonstrations. In addition, although the LLaMA-3.1-70B-Instruct model can already generate a concise logical form (*greater { row_0 ; population (january 1 , 2008) ; row_4 } = true*) and correct description, LogicTableLLaMA-3.1-8B can generate the logical form and description that are completely consistent with the reference answer at a fine-grained level. It further confirms that instruction-tuning the 8B-level LLM on our instruction dataset can enhance its logical reasoning ability on tables, even surpassing the 70B-level LLM. Overall, LogicTableLLaMA-3.1-8B with the 5-shot LogicToP is the optimal combination solution for the logical table-to-text generation task.

5 Conclusion

In this paper, we conducted the first in-depth exploration of the effectiveness of various LLMs in the logical table-to-text generation task. To simultaneously improve the logical reasoning ability of LLMs on tables and avoid the overhead caused by specific pre-training, we constructed a new table-related instruction dataset LogicTableInstruct for fine-tuning open-source LLMs. We introduced a novel reasoning framework termed LogicToP for controlled logical table-to-text generation to solve the problem of existing methods being unable to effectively perform logical reasoning in intricate non-chain structure scenarios. Our extensive experiments on various LLMs demonstrated that our proposed framework can effectively improve performance on this task, and our LogicTableLLaMA-3.1-8B model outperformed the state-of-the-art baseline on the Logic2Text dataset. We hope that the proposed method (LogicToP) can inspire other researchers in related fields.

Limitations

Our approach has three limitations: (1) We did not explore the impact of data ratios between different tasks during the table instruction-tuning phase; (2) Although we have found that using code data for continual pre-training can improve the model's performance on this task, we cannot achieve continual pre-training on LLMs at the 70B level due to our limited computing resources. (3) On the LogicNLG dataset, our task setting did not include standard logical forms in the input, resulting in high task difficulty. Although our method improved the logical reasoning ability of almost all tested LLMs, there was a significant gap in scores compared to methods based on especially pre-trained models in surface-level metrics.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Iñigo Alonso and Eneko Agirre. 2024. Automatic logical forms improve fidelity in table-to-text generation. *Expert Syst. Appl.*, 238(Part D):121869.

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.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. a. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Transactions on Machine Learning Research*.

Wenqing Chen, Jidong Tian, Yitian Li, Hao He, and Yaohui Jin. 2021. De-confounded variational encoder-decoder for logical table-to-text generation. In *Annual Meeting of the Association for Computational Linguistics*.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. b. Longlora: Efficient fine-tuning of long-context large language models. In *The Twelfth International Conference on Learning Representations*.

Zhiyu Chen, Wenhu Chen, Hanwen Zha, Xiyou Zhou, Yunkai Zhang, Sairam Sundaresan, and William Yang Wang. 2020b. Logic2text: High-fidelity natural language generation from logical forms. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2096–2111.

Shumin Deng, Jiacheng Yang, Hongbin Ye, Chuanqi Tan, Mosha Chen, Songfang Huang, Fei Huang, Huajun Chen, and Ningyu Zhang. 2023. LOGEN: few-shot logical knowledge-conditioned text generation with self-training. *IEEE ACM Trans. Audio Speech Lang. Process.*, 31:2124–2133.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, pages 10764–10799.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, et al. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*.

Hanxu Hu, Hongyuan Lu, Huajian Zhang, Wai Lam, and Yue Zhang. 2023. Chain-of-symbol prompting elicits planning in large langauge models. *CoRR*, abs/2305.10276.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdel rahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Annual Meeting of the Association for Computational Linguistics*.

Peng Li, Yeye He, Dror Yashar, Weiwei Cui, Song Ge, Haidong Zhang, Danielle Rifinski Fainman, Dongmei Zhang, and Surajit Chaudhuri. 2023. Table-gpt: Table-tuned gpt for diverse table tasks. *arXiv preprint arXiv:2310.09263*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Ao Liu, Haoyu Dong, Naoaki Okazaki, Shi Han, and Dongmei Zhang. 2022. Plog: Table-to-logic pretraining for logical table-to-text generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5531–5546.

Linyong Nan, Lorenzo Jaime Yu 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, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 6903–6917. Association for Computational Linguistics.

Xuefei Ning, Zinan Lin, Zixuan Zhou, Huazhong Yang, and Yu Wang. 2023. Skeleton-of-thought: Large language models can do parallel decoding. *CoRR*, abs/2307.15337.

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.

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, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

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-Dor, Dafna Sheinwald, Noam Slonim, and Michal Shmueli-Scheuer. 2022. Diversity enhanced table-to-text generation via type control. *ArXiv*, abs/2205.10938.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

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. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Weiwei Shi, Yubo Liu, Jie Wu, and Jianming Liao. 2022. Three-stage logical table-to-text generation based on type control. In *Proceedings of the 5th International Conference on Algorithms, Computing and Artificial Intelligence, ACAI 2022, Sanya, China, December 23-25, 2022*, pages 13:1–13:5. ACM.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv e-prints*, pages arXiv–2307.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.

Jiehui Wu and Mengshu Hou. 2024. Enhancing diversity for logical table-to-text generation with mixture of experts. *Expert Systems*, 41.

Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. 2024. Faithful logical reasoning via symbolic chain-of-thought. *arXiv e-prints*, pages arXiv–2405.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *CoRR*, abs/2305.10601.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. 2023. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.

Tianshu Zhang, Xiang Yue, Yifei Li, and Huan Sun. 2024. Tablellama: Towards open large generalist models for tables. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6024–6044.

Xueliang Zhao, Tingchen Fu, Lemao Liu, Lingpeng Kong, Shuming Shi, and Rui Yan. 2023a. Sortie: Dependency-aware symbolic reasoning for logical data-to-text generation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11247–11266.

Yilun Zhao, Zhenting Qi, Linyong Nan, Lorenzo Jaime Flores, and Dragomir Radev. 2023b. Loft: Enhancing faithfulness and diversity for table-to-text generation via logic form control. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 554–561.

Alex Zhuang, Ge Zhang, Tianyu Zheng, Xinrun Du, Junjie Wang, Weiming Ren, Stephen W Huang, Jie Fu, Xiang Yue, and Wenhu Chen. 2024. Structlm: Towards building generalist models for structured knowledge grounding. *arXiv preprint arXiv:2402.16671*.

A Parameters of Table Instruction-tuning

We provide detailed parameter settings for the table instruction-tuning in Table 4.

Parameter	LLaMA	Qwen2.5
batch size (per device)	3	3
gradient accumulation steps	2	8
learning rate	1.0e-4	1.0e-5
train epochs	3	4
lr scheduler type	cosine	cosine
warmup ratio	0.1	0.1
fp16	true	true
ddp timeout	1.8e8	1.8e8

Table 4: Detailed parameter settings for the table instruction-tuning. LLaMA: LLaMA-3.1-8B-Instruct. Qwen2.5: Qwen2.5-7B-Instruct.

Count
Definition: ***counting some rows in the table based on the values in one column, with the scope of all table rows or a subset of rows.***
An example of logic form:
eq { count { filter_eq { all_rows ; rank ; 20 } } ; 2 } = true

Superlative
Definition: ***Describing the maximum or minimum value in a column, with the scope of all table rows or a subset of rows. You may also talk about other columns in this row with the superlative value.***
An example of logic form:
eq { hop { argmax { all_rows ; number of viewers } ; show } ; 1966 world cup final } = true

Ordinal
Definition: ***Describing the n-th maximum or minimum value in a column, with the scope of all table rows or a subset of rows. You may also talk about other columns in this row with the n-th maximum or minimum value.***
An example of logic form:
eq { hop { nth_argmax { all_rows ; year ; 2 } ; album title } ; realism } = true

Comparative
Definition: ***Comparing two rows in the table, regarding their values in one column. You may also talk about other columns in these two rows.***
An example of logic form:
less { hop { filter_eq { all_rows ; title ; face / off } ; year } ; hop { filter_eq { all_rows ; title ; antz } ; year } } = true

Aggregation
Definition: ***Describing the sum or average value over a column, with the scope of all table rows or a subset of rows.***
An example of logic form:
round_eq { sum { filter_eq { all_rows ; place ; waddon } ; platforms } ; 4 } = true

Unique
Definition: ***Describing one unique row, regarding one column, with the scope of all table rows or a subset of rows. You may also talk about other columns in this unique row.***
An example of logic form:
and { only { filter_eq { all_rows ; host ; jack arute } } ; eq { hop { filter_eq { all_rows ; host ; jack arute } ; year } ; 2008 } } = true

Majority
Definition: ***Describing the majority values (most or all) over one column, with the scope of all table rows or a subset of rows.***
An example of logic form:
most_greater_eq { all_rows ; crowd ; 10000 } = true

Figure 5: Definitions and examples of seven logical types.

B Logical Type Definitions and Examples

As shown in Figure 5, we have provided the definitions of different logical types and examples of their linearized logic forms.

C Logical Type Statistics

As shown in Figure 6, we provide a visual view of the total number of different logical types in our LogicTableInstruct dataset.

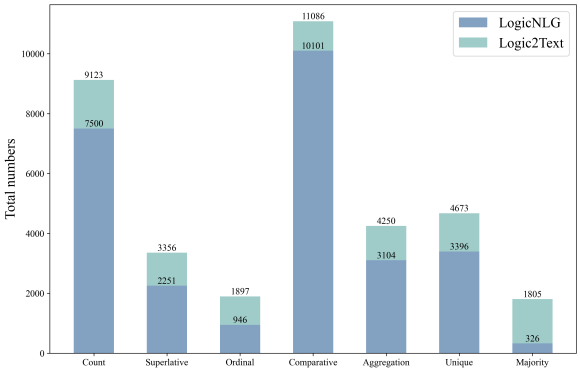

Figure 6: The total number of different logical types.

D The Prompt Format of LogicToP

See Figure 7.

E The Function Definitions List

See Table 5.

Prompt Format of LogicToP

<<System Role Setting>>
You are a scientific researcher with logical reasoning ability and comparative analysis ability.
<<Task Objectives and Key Definitions of Logical Types>>
This is a logical table-to-text generation task with highlighted cells, aiming to generate accurate and fluent natural language descriptions for the highlighted cells in the table.
Then, I will provide you with table_title, table_header, table_content, highlight_cells, and logical_type. Among them, "table_header" is the table header in list format; "table_content" consists of the table content as a list of lists.
"logical_type":
(1) Count: Counting some rows in the table based on the values in one column, with the scope of all table rows or a subset of rows.
(2) Superlative: Describing the maximum or minimum value in a column, with the scope of all table rows or a subset of rows. You may also talk about other columns in this row with the superlative value.
(3) Ordinal: Describing the n-th maximum or minimum value in a column, with the scope of all table rows or a subset of rows. You may also talk about other columns in this row with the n-th maximum or minimum value.
(4) Comparative: Comparing two rows in the table, regarding their values in one column. You may also talk about other columns in these two rows.
(5) Aggregation: Describing the sum or average value over a column, with the scope of all table rows or a subset of rows.
(6) Unique: Describing one unique row, regarding one column, with the scope of all table rows or a subset of rows. You may also talk about other columns in this unique row.
(7) Majority: Describing the majority values (most or all) over one column, with the scope of all table rows or a subset of rows.
<<Instruction for Generating natural language descriptions>>
Please generate semantically accurate natural language descriptions that match the highlighted cells based on the given table and logical type.
<<Instruction for Generating Logic Tree-of-Program>>
Before generating descriptive text, you need to perform logical reasoning first. Logical reasoning requires you to generate a logical form that matches the highlighted cells and the logical type based on the given table.
<<Instruction for In-Context Learning>>
Here are some examples: {*ICL_Content_LogicToP*}.
Please refer to the examples to process the following data: table_title:{*table_title*}, table_header:{*table_header*} , table_content:{*table_content*}, highlight_cells:{*highlight_cells*}, and logical_type:{*logical_type*}.

Figure 7: Prompt used for Logic Tree-of-Program (LogicToP). Each example in *ICL_Content_LogicToP* contains input, output, and the corresponding logical form of the output.

Name	Arguments	Output	Description
count	view	number	returns the number of rows in the view
only	view	bool	returns whether there is exactly one row in the view
hop	row, header string	object	returns the value under the header column of the row
and	bool, bool	bool	returns the boolean operation result of two arguments
max/min/avg/sum	view, header string	number	returns the max/min/average/sum of the values under the header column
nth_max/nth_min	view, header string	number	returns the n-th max/n-th min of the values under the header column
argmax/argmin	view, header string	row	returns the row with the max/min value in header column
nth_argmax/nth_argmin	view, header string	row	returns the row with the n-th max/min value in header column
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 argument 1 is greater/less than argument 2
diff	object, object	object	returns the difference between two arguments
filter_eq/not_eq	view, header string, object	view	returns the subview whose values under the header column is equal/not equal to argument 3
filter_greater/less	view, header string, object	view	returns the subview whose values under the header column is greater/less than argument 3
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 argument 3
filter_all	view, header string	view	returns the view itself for the case of describing the whole table
all_eq/not_eq	view, header string, object	bool	returns whether all the values under the header column are equal/not equal to argument 3
all_greater/less	view, header string, object	bool	returns whether all the values under the header column are greater/less than argument 3
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 argument 3
most_eq/not_eq	view, header string, object	bool	returns whether most of the values under the header column are equal/not equal to argument 3
most_greater/less	view, header string, object	bool	returns whether most of the values under the header column are greater/less than argument 3
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 argument 3

Table 5: Function definitions.