

STRUC-BENCH: Are Large Language Models Good at Generating Complex Structured Tabular Data?

Xiangru Tang^{♣*} Yiming Zong^{♡*} Jason Phang[◇] Yilun Zhao[♣] Wangchunshu Zhou[♣]

Arman Cohan[♣] Mark Gerstein[♣]

[♣] Yale University [♡] Zhejiang University [◇] New York University
xiangru.tang@yale.edu

Abstract

Despite the remarkable capabilities of Large Language Models (LLMs) like GPT-4, producing complex, structured tabular data remains challenging. Our study assesses LLMs’ proficiency in structuring tables and introduces a novel fine-tuning method, cognizant of data structures, to bolster their performance. We unveil STRUC-BENCH, a comprehensive benchmark featuring prominent LLMs (GPT-NeoX-20B, GPT-3.5, GPT-4, and Vicuna), which spans text tables, HTML, and LaTeX formats. Our proposed FORMATCOT aids in crafting format-specific instructions from the intended outputs to populate this benchmark. Addressing the gap in task-centered evaluation, we propose two innovative metrics, P-Score (Prompting Score) and H-Score (Heuristical Score), to more accurately gauge LLM performance. Our experiments show that applying our structure-aware fine-tuning to LLaMA-7B leads to substantial performance gains, outshining its LLM counterparts across most measures. In-depth error analysis and creating an ability map across six dimensions—coverage, formatting, reasoning, comprehension, pragmatics, and hallucination—highlight areas for future enhancements and suggest forthcoming research trajectories. Our code and models can be found at <https://github.com/gersteinlab/Struc-Bench>.

1 Introduction

Significant advancements have been made in various natural language processing tasks by Large Language Models (LLMs) (Brown et al., 2020; Scao et al., 2022; Ouyang et al., 2022; Muennighoff et al., 2022; OpenAI, 2023; Zhao et al., 2023a), especially in text generation tasks (Qin et al., 2023). The ability to output structured data, one of the key aspects of generative capability, has also attracted

great interest in previous studies (Wu et al., 2022; Zhao et al., 2023c,b; Zha et al., 2023).

Despite their advanced capabilities, LLMs have problems with generating complex structured tables, an indispensable skill for practical applications like coding copilot and automated report generation. This proficiency entails the organization of information from multifarious sources into coherent structures. Generating structured tables as outputs not only helps human understanding but also facilitates the automated data processing pipeline in autonomous language agents. Furthermore, generating structured tables can also serve as a critical preprocessing procedure for downstream tasks such as decision-making and knowledge extraction. However, the current landscape of LLM evaluation often neglects this aspect of table generation, which casts uncertainty on their full potential and utility in such scenarios. Our research seeks to thoroughly investigate these gaps.

First, there is a lack of systematic analysis and comprehensive benchmarks of the ability of LLMs to output complex structured tabular data. Previous efforts on evaluating LLMs (Qin et al., 2023; Ma et al., 2023) on structured data primarily centered around simple Information Extraction (IE) tasks: recognizing named entities, extracting relations, and detecting events. Here the goal of IE tasks is to gather the extracted data in a highly structured form (Zhong and Chen, 2020). Much earlier work was considerably more task-centric as opposed to LLM-centric. The focus was predominantly on generating structured data from text (text-to-data) tasks with pre-trained models (He et al., 2023; Rossiello et al., 2022; Whitehouse et al., 2023; Pietruszka et al., 2022) like BART (Lewis et al., 2019) and T5 (Raffel et al., 2020).

Second, there is a lack of evaluation metrics of structured tabular data generation. Existing benchmarks often rely on rudimentary objective metrics such as word overlap to measure the accuracy of

* Contributed equally.

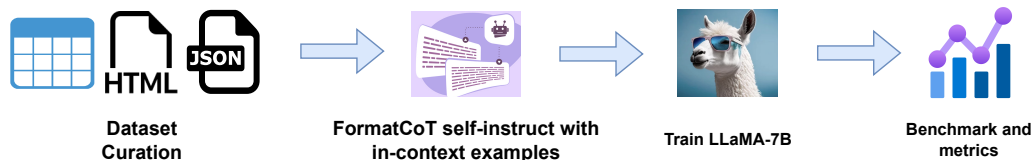


Figure 1: Overview of our workflow: We commenced by creating datasets of raw text tables, HTML tables, and \LaTeX tables. Subsequently, LLaMA-7B was trained using the training data constructed by FORMATCOT. Finally, our benchmarks validate the effectiveness of the current LLMs to generate such tables.

the content generated by the model (Li et al., 2023; Wu et al., 2022; Pietruszka et al., 2022). This may be insufficient for evaluating whether LLMs can generate structured output, as an ideal evaluation metric ought to also consider the format of generated content.

Third, there is a lack of methods to enhance the performance of current LLMs to better *follow natural language inputs and generate tabular outputs with the correct format*.

Our contributions can be encapsulated as outlined in Figure 1: (1) We introduce STRUC-BENCH, a benchmark specifically constructed for generating structured tabular data. (2) We evaluate popular LLMs on STRUC-BENCH using two proposed metrics, providing a comprehensive insight into the prevailing limitations and common error types. (3) We propose FORMATCOT to generate instruction tuning data, wherein we utilize GPT-3.5 to generate format instructions and then fine-tune LLaMA-7B model to follow these formats. The resulting impressive performance demonstrates that with FORMATCOT small models can indeed surpass the performance of a larger model in this particular task.

2 Problem Analysis and Benchmark

2.1 Problem Definition and Motivation

LLMs are tasked with generating complex structured tables, a process that involves understanding both the content and the specific format requirements, such as LaTeX syntax. This task extends beyond simple text generation as it demands precision not just in content creation but also in adhering to a detailed and precise structural format. Specially, we aim to convert unstructured textual data into structured tabular data, by extracting necessary contents from text and following the required structure or format.

2.2 Problem Analysis

To assess LLMs’ capability to convert textual descriptions to structured tables, we utilized the RotoWire dataset (Wiseman et al., 2017), originally

a table-to-text dataset, in reverse as a text-to-table task. After ensuring that the descriptions contained adequate information for table generation through a review of 20 samples, we found significant limitations in the performance of GPT-3.5 and GPT-4, especially when dealing with complex structures as detailed in Appendix A.

When put to the test of generating data in precise formats, such as tables, both GPT-3.5 and GPT-4, despite their advanced capabilities, frequently erred, as evidenced by a systematic MTurk human annotation study (refer to Appendix B). The types of errors, categorized into ‘Element Errors’, ‘Element Format Errors’, ‘Structure Error’, and ‘Structure Naming Errors’, are quantified in Figure 2.

A mere 3% of GPT-3.5’s outputs were fully accurate, with GPT-4 only slightly better at 9%. These results suggest design limitations within the GPT architecture, which, although effective at mimicking language patterns, falter in tasks requiring sustained structural coherence over longer sequences.

2.3 Benchmark Construction

We begin by selectively sourcing tables larger than 3x3 from the RotoWire (Wiseman et al., 2017) dataset to present a baseline of complexity. Then, to broaden our dataset diversity across various domains, from The Stack (Kocetkov et al., 2022), which includes GitHub code in 358 programming languages from the BigCode project, we first select LaTeX and HTML formats. Further refining our dataset, we extract elements relevant to table representations to ensure focused complexity and relevance to our structured data generation task. An example of our benchmark is shown in Figure 4.

Table 1 gives statistics for the Rotowire dataset and our constructed datasets. Then we evaluate 4 popular LLMs, including GPT-NeoX-20B (Black et al., 2022), GPT-3.5, GPT-4, and Vicuna-13B (Chiang et al., 2023). For LaTeX and HTML data without paired text, we harness GPT-3.5 to construct synthetic descriptions to be utilized as input. To guarantee the quality of our

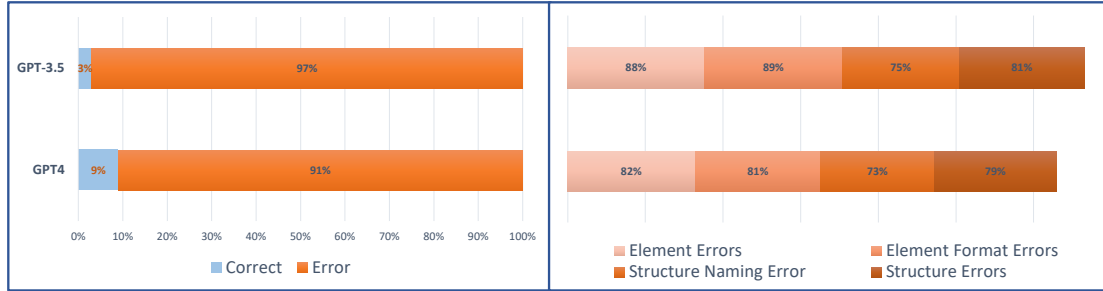


Figure 2: Error analysis by human annotation. Some error types are explained in Appendix A.

benchmark, we sample 50 tables for each format to ensure the correctness of the descriptions. Initially, we achieved a satisfaction rate of 76%. However, upon incorporating a manual interpretation template (e.g. tab names for HTML) tailored to each format (Appendix D), our satisfaction rate improved significantly, reaching 96%. For example, HTML tables possess their unique tags and structure, conforming faithfully to the syntax rules of HTML language.

Dataset	# Train	# Test	Format	Rows & Columns
STRUC-BENCH Table	3.4k	728	Raw tex	7.26 & 8.75
STRUC-BENCH L ^A T _E X	5.3k	500	L ^A T _E X	2.75 & 4.47
STRUC-BENCH HTML	5.4k	499	HTML	5.50 & 3.54

Table 1: STRUC-BENCH data statistics. The number of Rows & Columns has been averaged.

3 Methodology

3.1 Data Generation

As shown in Figure 3, we propose FORMATCOT with GPT-3.5, a self-instruct method to generate {data, instruction} pairs for fine-tuning purposes. Specifically, our prompt of FORMATCOT involves guiding models to accurately describe and interpret the format elements presented in the output

table, inspired by Wang et al. (2023b) in the summarization task. To verify the effectiveness of our proposed FORMATCOT, we conduct an ablation study in Appendix F.

3.2 Instruction Tuning

We introduce an instruction tuning approach designed specifically to enhance LLMs’ abilities in generating structured text (Touvron et al., 2023; Patil et al., 2023). Specifically, we combine GPT-3.5-generated format descriptions of output tables and the original text input as the new input of LLaMA fine-tuning. In other words, we start with GPT-3.5 processing table data and synthesizing comprehensive format instructions. The LLaMA model is then fine-tuned on these enriched instructions we generate. This approach simulates a user-agent interaction where GPT-3.5 effectively fetches and consolidates table information, conversationally instructing LLaMA for the final text generation, outlined in Figure 3.

3.3 Evaluation Metrics

Assessing the accuracy of generated tables against ground truth is complex due to the variability in formatting, like HTML. An ideal evaluation metric needs to discern substantial data discrepancies

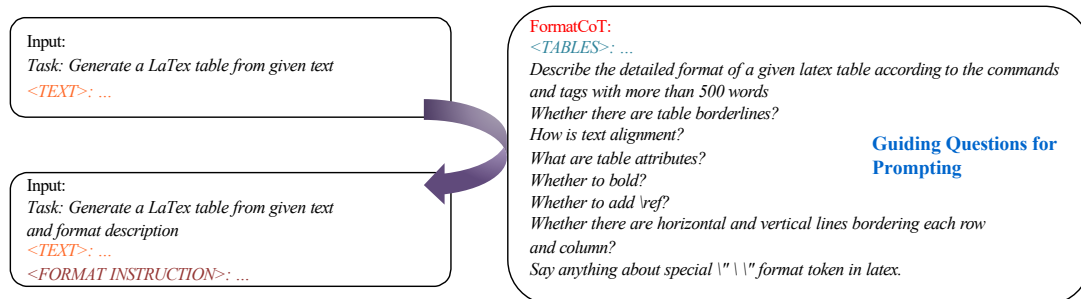


Figure 3: The upper-left corner box represents the original input, which notably lacks a description of the format. To explicitly instruct the model on format understanding, we employ the FORMATCOT located on the right, which produces the <FORMAT INSTRUCTION>. The lower-left box illustrates what the input for LLaMA fine-tuning looks like after passing through FORMATCOT. <TEXT> provides a descriptive text for the expected table output (original input), <TABLE> serves as a reference table (output), and the <FORMAT INSTRUCTION> is a format guideline generated through FORMATCOT (added into input). Detailed prompts are displayed in Appendix D.1.

while disregarding trivial formatting variations.

We propose to break down the similarity of two tables into two coarse components: *content* and *format*. In scoring content similarity, we attempt to parse *content* out of the data within the table cells, and compute the similarity. This similarity is computed between the generated and ground-truth table cells by commonly used similarity metrics. In scoring format similarity, we place higher emphasis on components such as the number of columns and rows, cell alignment, and the table caption. We find that these two scores allow us to perform a more involved analysis of where predicted and ground-truth tables differ. The implementation of these two scores can be found in Appendix C.

3.3.1 P-Score

We take two approaches to score each metric. First, we perform model-based evaluation, querying GPT-3.5 with both tables and having it score the similarity of content and format separately. Following Wang et al. (2023a), we prompt the model to perform Chain-of-Thought (Wei et al., 2023) reasoning before outputting its scores, and we query the model with the predicted and ground-truth tables in both orders and average the scores. We report these as the *P-Score* (Prompting Score).

3.3.2 H-Score

In addition, we also implement hand-crafted scoring functions to score the similarity of the tables. Since the tables can be presented in different formats, we implement several heuristics to normalize the tables and to compute their similarity. We use

an average of Levenshtein distance and the Ratcliff/Obershelp similarity metric to compute the similarities between strings or data structures. These heuristically normalized metrics are reported as the *H-Score* (Heuristical Score). The analysis can be found in Appendix A.3.

4 Experiments

4.1 Basic Settings

For metrics, we use SacreBLEU, ROUGE-L, BERTScore, BARTScore, and BLEURT metrics as they are all classical metrics to evaluate text similarity, as well as two proposed metrics: P-Score and H-score. We evaluate the following models: GPT-NeoX-20B, GPT-3.5, GPT-4, Vicuna-13B, LLaMA-7B, and our finetuning LLaMa-7B. GPT-NeoX-20B, GPT-3.5 and GPT-4 represent the state-of-art performance of current LLMs and Vicuna-13B is another version finetuned on LLaMA, which can reach 90% of the capacity of GPT-3.5. We think these models are strong enough to be persuasive. For the first 4 models, we simply call their APIs from OpenAI or HuggingFace to generate results without further finetuning. In our dataset, each item consists of three parts: instruction, input, and output. When generating results, we put each item’s instruction and input together as the final input to models. During inference, the user provides the prompt in natural language, this can be for a simple task (e.g., “please generate a table given by the following information and format”). During the inference process, we provide the model with a natural language prompt to describe the format

Model	SacreBLEU	ROUGE-L	BERTScore	BARTScore	BLEURT	Content P-Score	Format P-Score	Content H-Score	Format H-Score
<i>Tables from Raw Text</i>									
GPT-NeoX-20B	35.24	55.78	68.91	-2.34	33.51	3.86	6.10	0.50	-1.32
GPT-3.5	56.92	70.97	91.35	-1.68	36.85	6.19	8.16	0.52	-1.27
GPT-4	68.13	75.44	94.89	-0.99	55.24	6.88	8.30	0.85	0.53
Vicuna-13B	40.12	50.77	75.21	-2.05	40.02	4.07	6.33	0.55	-1.38
Ours-7B	90.6	88.98	98.54	-0.69	66.07	7.69	8.60	1.65	3.61
<i>w.o. finetune</i>	9.9	36.56	81.63	-2.50	70.24	4.58	6.00	0.51	-1.01
<i>LaTeX</i>									
GPT-NeoX-20B	45.92	65.10	76.09	-2.05	40.87	7.23	7.02	0.56	0.72
GPT-3.5	56.94	75.99	86.25	-1.30	42.89	8.22	8.41	0.99	1.27
GPT-4	78.15	85.34	88.07	-1.09	67.11	8.78	8.81	1.10	1.35
Vicuna-13B	50.80	69.48	80.44	-1.07	36.74	7.70	8.10	0.78	1.06
Ours-7B	89.13	88.99	98.55	-0.69	66.07	8.94	9.05	1.14	1.52
<i>w.o. finetune</i>	47.24	70.89	73.27	-2.13	38.13	7.10	6.98	0.51	0.69
<i>HTML</i>									
GPT-NeoX-20B	60.36	72.13	86.88	-1.59	30.06	8.42	8.94	0.81	0.92
GPT-3.5	73.80	85.19	96.76	-1.46	34.81	9.11	9.35	1.10	2.15
GPT-4	79.25	85.95	97.22	-1.31	41.59	9.17	9.62	1.15	2.29
Vicuna-13B	58.75	70.37	88.65	-1.58	31.11	8.55	8.88	0.79	0.93
Ours-7B	77.50	86.08	96.25	-1.30	42.89	9.20	9.70	1.18	2.49
<i>w.o. finetune</i>	65.30	78.24	88.12	-1.57	32.78	8.22	8.81	0.92	0.96

Table 2: Automated evaluation results on the test set, involving five types of previous metrics and four proposed ones. *w.o. finetune* means that we also compared the performance of our model without finetuning as an ablation study. ‘Ours-7B’ is finetuned LLaMA.

You

Generate tables according to the following description:
Input text: The Atlanta Hawks (46 - 12) beat the Orlando Magic (19 - 41) 95 - 88 on Friday. Al Horford had a good all-around game, putting up 17 points, 13 rebounds, four assists and two steals in a tough matchup against Nikola Vucevic. Kyle Korver was the lone Atlanta starter not to reach double figures in points. Jeff Teague bounced back from an illness, he scored 17 points to go along with seven assists and two steals. After a rough start to the month, the Hawks have won three straight and sit atop the Eastern Conference with a nine game lead on the second place Toronto Raptors. The Magic lost in devastating fashion to the Miami Heat in overtime Wednesday. They blew a seven point lead with 43 seconds remaining and they might have carried that with them into Friday's contest against the Hawks. Vucevic led the Magic with 21 points and 15 rebounds. Aaron Gordon (ankle) and Evan Fournier (hip) were unable to play due to injury. The Magic have four teams between them and the eighth and final playoff spot in the Eastern Conference. The Magic will host the Charlotte Hornets on Sunday, and the Hawks will take on the Heat in Miami on Saturday.

ChatGPT

Based on the provided input text, here are the tables you requested:

Table 1: Team Stats

Team	Losses	Total points	Points in 4th quarter	Wins
Hawks	12	95	26	46
Magic	41	88	22	19

Table 2: Player Stats

Player	Assists	Points	Total rebounds	Steals
Al Horford	4	17	13	2
Kyle Korver	-	-	-	-
Jeff Teague	7	17	-	2
Nikola Vucevic	-	21	15	-

Let me know if there are any adjustments you'd like to make!

Figure 4: An exemplification of our benchmark. The input is made up of the instruction and the input text, whereas the output aims to present the target table. Notably, there are some inaccuracies in the predicted output; for instance, ‘Points in 4th quarter’ under ‘Hawks’ should be vacant, and correspondingly, ‘Points in 4th quarter’ for ‘Magic’ should be 21.

and content of our task, as well as the expected response.

4.2 Human Evaluation

Table 3 displays human evaluation results on two proposed metrics with instance-level Pearson correlation, reflecting a purposeful design that caters to the specific demands of structured output assessment. We engaged five undergraduate students to annotate 200 examples focusing on content and format quality. Equipped with the input description (with reference appended) and generated outputs, they scored each aspect on a 10-point scale. Both the P-score and H-score showcase a significant correlation with human judgment, indicating their relative robustness and effectiveness in this evaluation space. This level of correlation, which surpasses that of many prior meta-evaluation efforts (Fabri et al., 2020; Tang et al., 2021), reinforces the value of our metrics and addresses concerns about their ability to reliably reflect human evaluation.

Additionally, we evaluated well-known metrics including ROUGE-L, BERTScore, BARTScore, and BLEURT. Limited space precluded a full discussion, yet our Content P-score showed the best instance-level correlation.

Metrics	Content Correlation	Format Correlation
Content P-score	0.5301	-
Format P-score	-	0.3812
Content H-score	0.1059	-
Format H-score	-	0.3021

Table 3: Human evaluation results.

4.3 Results

Table 2 provides a comparative analysis of different LLMs based on several metrics. For ‘Tables from Raw Text’, the Ours-7B outperforms the other models in every metric. Interestingly, without fine-tuning, the performance drops significantly, particularly in SacreBLEU, ROUGE-L, and BERTScore. The results for ‘LaTeX’ reveal a similar trend and in the ‘HTML’ category, GPT-4 scores the highest in SacreBLEU and BERTScore. However, these differences are slight and our 7B model comes out on top for the rest of the metrics. The results demonstrate that our approach exhibits superior performance, highlighting the efficacy of fine-tuning smaller models in surpassing much larger models. Moreover, we delve into an analysis based on our Mturk annotation, attributing observed shortcomings to several error types. And we present an ability map in Figure 5 and Appendix E.

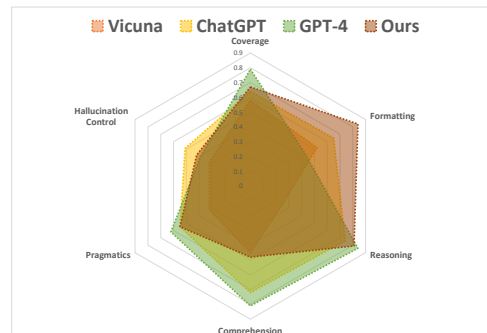


Figure 5: Visualization of LLMs’ capability.

5 Conclusion

In summary, our study provides a thorough analysis of LLMs’ challenges in structured table generation, introduces novel evaluation metrics, and assembles a specific benchmark covering a range of data types. We pinpoint key issues including content fidelity, format adherence, numerical reasoning, and management of extensive tables.

6 Limitations

Although we present a comprehensive analysis, the exploration of LLMs in structured text generation presented in this paper has several limitations:

Investigating Optimal Format for Tabular Representation In this study, we did not investigate which table formats are most effective. Different presentations of the same information can be reasonable, and **table normalization** strategies, such as determining the best way to tabulate given facts or how to interconnect multiple tables, remain unexplored. Future research could engage in the study of table normalization to ascertain optimal strategies for tabular data structuring and representation.

Domain-Specific Benchmark Development While we’ve made strides in constructing benchmarks for structured text generation, it may be beneficial to develop benchmarks that cater to specific domains. Different fields might have unique structural requirements and understanding these nuances can significantly improve the models’ applicability across diverse contexts.

Expand the Range of Datasets There are endless data types and sources that can be explored. Incorporating a broader variety of datasets could expose the models to an even wider range of structural formats, ultimately enhancing their overall performance.

Enhancing Numerical Reasoning Capabilities Our study identified inadequate numerical reasoning as one of the challenges faced by LLMs. Investigating techniques to bolster numerical reasoning in these models could lead to significant improvements in their performance.

Developing Advanced Methods While our structure-aware instruction tuning method showed promising results, more sophisticated techniques could be developed. For instance, future work could explore ways of incorporating more explicit structural information into the model or developing methods that allow the model to learn structural patterns more effectively.

Exploring Multimodal LLMs As LLMs continue to evolve, there are opportunities to explore multimodal models that can process and generate both text and other forms of data, such as sound or images (Kamigaito et al., 2023), in a structured manner.

References

- Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. 2022. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.
- Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. See <https://vicuna.lmsys.org> (accessed 14 April 2023).
- Alexander R Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2020. Summeval: Re-evaluating summarization evaluation. *arXiv preprint arXiv:2007.12626*.
- Yuxin He, Jingyue Hu, and Buzhou Tang. 2023. Revisiting event argument extraction: Can eae models learn better when being aware of event co-occurrences? *arXiv preprint arXiv:2306.00502*.
- Hidetaka Kamigaito, Katsuhiko Hayashi, and Taro Watanabe. 2023. Table and image generation for investigating knowledge of entities in pre-trained vision and language models. *arXiv preprint arXiv:2306.02115*.
- Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro von Werra, and Harm de Vries. 2022. The stack: 3 tb of permissively licensed source code. *Preprint*.
- Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.
- Tong Li, Zhihao Wang, Liangying Shao, Xuling Zheng, Xiaoli Wang, and Jinsong Su. 2023. A sequence-to-sequence&set model for text-to-table generation. *arXiv preprint arXiv:2306.00137*.
- Yubo Ma, Yixin Cao, YongChing Hong, and Aixin Sun. 2023. Large language model is not a good few-shot information extractor, but a good reranker for hard samples! *arXiv preprint arXiv:2303.08559*.
- Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey

- Schoelkopf, et al. 2022. Crosslingual generalization through multitask finetuning. *arXiv preprint arXiv:2211.01786*.
- OpenAI. 2023. [Gpt-4 technical report](#).
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.
- Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*.
- Michał Pietruszka, Michał Turski, Łukasz Borchmann, Tomasz Dwojak, Gabriela Pałka, Karolina Szyndler, Dawid Jurkiewicz, and Łukasz Garncarek. 2022. Stable: Table generation framework for encoder-decoder models. *arXiv preprint arXiv:2206.04045*.
- Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. 2023. Is chatgpt a general-purpose natural language processing task solver? *arXiv preprint arXiv:2302.06476*.
- 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. *The Journal of Machine Learning Research*, 21(1):5485–5551.
- Gaetano Rossiello, Faisal Chowdhury, Nandana Mihindukulasooriya, Owen Corneec, and Alfio Gliozzo. 2022. Knowgl: Knowledge generation and linking from text. *arXiv preprint arXiv:2210.13952*.
- Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.
- Xiangru Tang, Alexander R Fabbri, Ziming Mao, Griffin Adams, Borui Wang, Haoran Li, Yashar Mehdad, and Dragomir Radev. 2021. Investigating crowdsourcing protocols for evaluating the factual consistency of summaries. *arXiv preprint arXiv:2109.09195*.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. [Llama: Open and efficient foundation language models](#).
- Peiyi Wang, Lei Li, Liang Chen, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023a. [Large language models are not fair evaluators](#).
- Yiming Wang, Zhuosheng Zhang, and Rui Wang. 2023b. Element-aware summarization with large language models: Expert-aligned evaluation and chain-of-thought method. *arXiv preprint arXiv:2305.13412*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. [Chain-of-thought prompting elicits reasoning in large language models](#).
- Chenxi Whitehouse, Clara Vania, Alham Fikri Aji, Christos Christodoulopoulos, and Andrea Pierleoni. 2023. Webie: Faithful and robust information extraction on the web. *arXiv preprint arXiv:2305.14293*.
- Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in data-to-document generation. *arXiv preprint arXiv:1707.08052*.
- Xueqing Wu, Jiacheng Zhang, and Hang Li. 2022. [Text-to-table: A new way of information extraction](#). In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2518–2533, Dublin, Ireland. Association for Computational Linguistics.
- Liangyu Zha, Junlin Zhou, Liyao Li, Rui Wang, Qingyi Huang, Saisai Yang, Jing Yuan, Changbao Su, Xiang Li, Aofeng Su, et al. 2023. Tablegpt: Towards unifying tables, nature language and commands into one gpt. *arXiv preprint arXiv:2307.08674*.
- Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023a. A survey of large language models. *arXiv preprint arXiv:2303.18223*.
- Yilun Zhao, Haowei Zhang, Shengyun Si, Linyong Nan, Xiangru Tang, and Arman Cohan. 2023b. Large language models are effective table-to-text generators, evaluators, and feedback providers. *arXiv preprint arXiv:2305.14987*.
- Yilun Zhao, Chen Zhao, Linyong Nan, Zhenting Qi, Wenlin Zhang, Xiangru Tang, Boyu Mi, and Dragomir Radev. 2023c. Robut: A systematic study of table qa robustness against human-annotated adversarial perturbations. *arXiv preprint arXiv:2306.14321*.
- Zexuan Zhong and Danqi Chen. 2020. A frustratingly easy approach for entity and relation extraction. *arXiv preprint arXiv:2010.12812*.

A Analysis with Examples

A.1 Example Table A

The main difference between the reference tables and the tables generated by GPT-3.5 and GPT4, shown in figure 6, is in the completeness and precision of the data provided.

In the reference tables, all relevant data is fully represented: For the teams (Table 1), each team has a precise number or percentage for every statistic. Similarly, for the players (Table 2), each player has a definite number for every statistic, including minutes played in the format "mm:ss".

In contrast, the generated tables show data that is incomplete and imprecise. For GPT-3.5 generated one, the team statistics table has some statistics missing, as represented by empty cells, and some are not presented as percentages. The player statistics table also has missing data similarly, and it lacks the "minutes played" statistics entirely. For instance, in the 'team' table, the "Percentage of field goals" column for the Suns is missing. Similarly, in the 'player' table, many key statistics such as "3-pointers attempted", "3-pointers made", "Field goals attempted", "Field goals made", and "Minutes played" are missing for various players. Regarding the format, we observe a lot of format errors. For example, the 'Percentage of field goals' column for Grizzlies is represented as "50" instead of "50.0%". Moreover, the 'Wins' column for the Suns is represented as "3" instead of "0". This misrepresentation can lead to significant misunderstanding of the data. The 'Player' table also has format errors. For instance, the 'Minutes played' column is missing the time format (i.e., "00:00"). On the other hand, the reference tables adhere to a standard format. Percentage data is represented with a '%' sign, time data uses the '00:00' format, and numeric data correctly represents each statistic.

For Vicuna-13B results shown in figure 7, although it has the correct format for both tables, there are still many element errors. For instance, the 'team' table has wrong statistics such as "Losses" and "Win" for the Suns. Besides, in the 'player' table, many cells shouldn't have data. However, they have, which is a mistake. Some cells like Isaiah Thomas's and Eric Bledsoe's 'Assists' should be 2 and 4, but they are not in the Vicuna-13B 'player' table. Similarly, LLaMA2-7B results, have the same element errors in the 'team' table and worse errors in the 'player' table. It fills all cells, many of which should be none. As for some

cells that should have data, their data are wrongly filled in like Eric Bledsoe's 'Assists' and 'Field goals made'.

The Grizzlies (50) used a strong second half to outlast the Suns (3 - 2) 102 - 91 in Phoenix on Wednesday night. Memphis found itself behind six at halftime but outscored Phoenix 30 - 19 in the third quarter and 26 - 20 in the final period. The Grizzlies shot 50 percent from the field, led by strong performances from Courtney Lee and Mike Conley. Lee scored 22 points (9 - 14 FG, 4 - 5 3Pt), while Conley led all scorers with 24 (9 - 14 FG, 3 - 4 3Pt) and 11 assists. Marc Gasol added 18 points, six assists, and five rebounds. The Suns, who beat the Lakers 112 - 106 on Tuesday, were paced by 23 points (9 - 12 FG), five rebounds and four assists from Eric Bledsoe. It was a quiet night for Goran Dragic, who scored just six points in 26 minutes. The third member of the backcourt trio, Isaiah Thomas, had 15 points and two assists off the bench, while Markieff Morris added 20 points and five rebounds. The Grizzlies out-rebounded Phoenix 37 - 35 and outscored the Suns in the paint 46 - 32. Memphis also registered 25 assists compared to only 13 on 32 field goals - for the Suns. Memphis now heads to Oklahoma City to take on the Thunder on Friday. Phoenix, meanwhile, hosts the Kings on Friday.

Table 1: Team Summary Reference

Team	Number of team assists	Percentage of field goals	Losses	Total points	Points in 3rd quarter	Points in 4th quarter	Rebounds	Wins
Suns	13	-	2	91	19	20	35	3
Grizzlies	25	50	-	102	30	26	37	-

GPT-3.5

Team	Number of team assists	Percentage of field goals	Losses	Total points	Points in 3rd quarter	Points in 4th quarter	Rebounds	Wins
Suns	13	47.7%	2	91	19	20	35	0
Grizzlies	25	50.0%	0	102	30	26	37	1

GPT-4

Team	Number of team assists	Percentage of field goals	Losses	Total points	Points in 3rd quarter	Points in 4th quarter	Rebounds	Wins	points in the paint
Grizzlies	25	50.0%	-	102	30	26	37	0	46
Suns	13	-	2	91	19	20	35	3	32

Table 2: Player Statistics Reference

Player	Assists	3-pointers attempted	3-pointers made	Field goals attempted	Field goals made	Minutes played	Points	Total rebounds
Marc Gasol	6	-	-	-	-	-	18	5
Courtney Lee	-	5	4	14	9	-	22	-
Mike Conley	11	4	3	14	9	-	24	-
Markieff Morris	-	-	-	-	-	-	20	5
Goran Dragic	-	-	-	-	-	26	6	-
Eric Bledsoe	4	-	-	12	9	-	23	5
Isaiah Thomas	2	-	-	-	-	-	15	-

GPT-3.5

Player	Assists	3-pointers attempted	3-pointers made	Field goals attempted	Field goals made	Minutes played	Points	Total rebounds
Marc Gasol	6	0	0	12	8	35:00	18	5
Courtney Lee	1	5	4	14	9	34:00	22	2
Mike Conley	11	4	3	14	9	36:00	24	0
Markieff Morris	1	5	2	14	8	34:00	20	5
Goran Dragic	4	2	1	12	9	26:00	6	5
Eric Bledsoe	4	3	1	12	9	34:00	23	5
Isaiah Thomas	2	5	2	11	5	26:00	15	0

GPT-4

Player	Assists	3-pointers attempted	3-pointers made	Field goals attempted	Field goals made	Minutes played	Points	Total rebounds
Marc Gasol	6	-	-	-	-	-	18	5
Courtney Lee	-	5	4	14	9	-	22	-
Mike Conley	11	4	3	14	9	-	24	-
Markieff Morris	-	-	-	-	-	-	20	5
Goran Dragic	-	-	-	-	-	26	6	-
Eric Bledsoe	4	-	-	12	9	-	23	5
Isaiah Thomas	2	-	-	-	-	-	15	-

Figure 6: Examples of generating tables with GPT-3.5 and GPT-4 based on the input text. The generated results contain a large number of errors, including format errors and content errors.

The Grizzlies (50) used a strong second half to outlast the Suns (3 - 2) 102 - 91 in Phoenix on Wednesday night. Memphis found itself behind six at halftime but outscored Phoenix 30 - 19 in the third quarter and 26 - 20 in the final period. The Grizzlies shot 50 percent from the field, led by strong performances from Courtney Lee and Mike Conley. Lee scored 22 points (9 - 14 FG, 4 - 5 3Pt), while Conley led all scorers with 24 (9 - 14 FG, 3 - 4 3Pt) and 11 assists. Marc Gasol added 18 points, six assists, and five rebounds. The Suns, who beat the Lakers 112 - 106 on Tuesday, were paced by 23 points (9 - 12 FG), five rebounds and four assists from Eric Bledsoe. It was a quiet night for Goran Dragic, who scored just six points in 26 minutes. The third member of the backcourt trio, Isaiah Thomas, had 15 points and two assists off the bench, while Markieff Morris added 20 points and five rebounds. The Grizzlies out-rebounded Phoenix 37 - 35 and outscored the Suns in the paint 46 - 32. Memphis also registered 25 assists compared to only 13 - on 32 field goals - for the Suns. Memphis now heads to Oklahoma City to take on the Thunder on Friday. Phoenix, meanwhile, hosts the Kings on Friday.

Table 1: Team Summary [Reference](#)

Team	Number of team assists	Percentage of field goals	Losses	Total points	Points in 3rd quarter	Points in 4th quarter	Rebounds	Wins
Suns	13	-	2	91	19	20	35	3
Grizzlies	25	50	-	102	30	26	37	-

Vicuna-13B

Team	Number of team assists	Percentage of field goals	Losses	Total points	Points in 3rd quarter	Points in 4th quarter	Rebounds	Wins
Suns	13	-	3	91	19	20	35	5
Grizzlies	25	50.0%	-	102	30	26	37	7

LLaMA2-7B

Team	Number of team assists	Percentage of field goals	Losses	Total points	Points in 3rd quarter	Points in 4th quarter	Rebounds	Wins
Suns	13	40.6%	2	91	19	20	35	0
Grizzlies	25	50.0%	1	102	30	26	37	1

Table 2: Player Statistics [Reference](#)

Player	Assists	3-pointers attempted	3-pointers made	Field goals attempted	Field goals made	Minutes played	Points	Total rebounds
Marc Gasol	6	-	-	14	9	32	18	5
Courtney Lee	-	4	5	14	9	32	22	-
Mike Conley	11	3	4	14	9	32	24	-
Markieff Morris	-	2	1	12	6	26	20	5
Goran Dragic	-	2	0	12	6	26	6	-
Eric Bledsoe	-	1	1	12	9	26	23	5
Isaiah Thomas	-	1	0	8	4	18	15	2

Vicuna-13B

Player	Assists	3-pointers attempted	3-pointers made	Field goals attempted	Field goals made	Minutes played	Points	Total rebounds
Marc Gasol	6	-	-	-	-	-	18	5
Courtney Lee	-	5	4	14	9	-	22	-
Mike Conley	11	4	3	14	9	-	24	-
Markieff Morris	-	-	-	-	-	-	20	5
Goran Dragic	-	-	-	-	-	26	6	-
Eric Bledsoe	4	-	-	12	9	-	23	5
Isaiah Thomas	2	-	-	-	-	-	15	-

LLaMA2-7B

Player	Assists	3-pointers attempted	3-pointers made	Field goals attempted	Field goals made	Minutes played	Points	Total rebounds
Marc Gasol	6	3	1	9	4	34	18	5
Courtney Lee	3	5	4	14	9	36	22	3
Mike Conley	11	4	3	14	9	36	24	3
Markieff Morris	2	2	0	12	5	31	20	5
Goran Dragic	4	3	1	12	6	26	6	2
Eric Bledsoe	5	2	1	12	7	35	23	4
Isaiah Thomas	2	2	1	11	4	23	15	2

Figure 7: Examples of generating tables with Vicuna-13B and LLaMA2-7B based on the input text, the generated results contain a large number of errors, including format errors and content errors.

The Grizzlies (50) used a strong second half to outlast the Suns (3 - 2) 102 - 91 in Phoenix on Wednesday night. Memphis found itself behind six at halftime but outscored Phoenix 30 - 19 in the third quarter and 26 - 20 in the final period. The Grizzlies shot 50 percent from the field, led by strong performances from Courtney Lee and Mike Conley. Lee scored 22 points (9 - 14 FG, 4 - 5 3Pt), while Conley led all scorers with 24 (9 - 14 FG, 3 - 4 3Pt) and 11 assists. Marc Gasol added 18 points, six assists, and five rebounds. The Suns, who beat the Lakers 112 - 106 on Tuesday, were paced by 23 points (9 - 12 FG), five rebounds and four assists from Eric Bledsoe. It was a quiet night for Goran Dragic, who scored just six points in 26 minutes. The third member of the backcourt trio, Isaiah Thomas, had 15 points and two assists off the bench, while Markieff Morris added 20 points and five rebounds. The Grizzlies out - rebounded Phoenix 37 - 35 and outscored the Suns in the paint 46 - 32. Memphis also registered 25 assists compared to only 13 - on 32 field goals - for the Suns. Memphis now heads to Oklahoma City to take on the Thunder on Friday. Phoenix, meanwhile, hosts the Kings on Friday.

Table 1: Team Summary Reference

Team	Number of team assists	Percentage of field goals	Losses	Total points	Points in 3rd quarter	Points in 4th quarter	Rebounds	Wins
Suns	13	-	2	91	19	20	35	3
Grizzlies	25	50	-	102	30	26	37	-

GPT-4

Team	Half-Time Score	Final Score	Field Goals (%)	3Pt (%)	Rebounds	Assists	Points in the Paint
Grizzlies	46	102	50	N/A	37	25	46
Suns	52	91	N/A	N/A	35	13	32

Table 2: Player Statistics Reference

Player	Team	Points	Assists	Rebounds	Field Goals (FG)	Three-Points (3Pt)
Courtney Lee	Grizzlies	22	N/A	N/A	9 - 14	4 - 5
Mike Conley	Grizzlies	24	11	N/A	9 - 14	3 - 4
Marc Gasol	Grizzlies	18	6	5	N/A	N/A
Eric Bledsoe	Suns	23	4	5	9 - 12	N/A
Goran Dragic	Suns	6	N/A	N/A	N/A	N/A
Isaiah Thomas	Suns	15	2	N/A	N/A	N/A
Markieff Morris	Suns	20	N/A	5	N/A	N/A

GPT-4

Player	Assists	3-pointers attempted	3-pointers made	Field goals attempted	Field goals made	Minutes played	Points	Total rebounds
Marc Gasol	6	-	-	-	-	-	18	5
Courtney Lee	-	5	4	14	9	-	22	-
Mike Conley	11	4	3	14	9	-	24	-
Markieff Morris	-	-	-	-	-	-	20	5
Goran Dragic	-	-	-	-	-	26	6	-
Eric Bledsoe	4	-	-	12	9	-	23	5
Isaiah Thomas	2	-	-	-	-	-	15	-

Figure 8: Using GPT-4 to generate a table based on the input text without FORMATCOT, the generated results contain a large number of errors, including format errors and content errors.

A.2 Error Type

Structure Errors: These errors pertain to the structural integrity of the generated tables. Specifically, they include instances where there are excess or missing rows or columns in comparison to the correct table structure. For instance, in figure 8 GPT4 generated result has missing columns like “Win” and “Losse” in the ‘team’ table.

Structure Naming Errors: This category captures errors related to the naming conventions used for rows or columns. Any discrepancies in a row or column names between the generated and correct table are flagged as structure naming errors. For instance, in figure 8, the GPT-4 generated result has wrong column names like “Half-Time Score” in the ‘team’ table.

Element Errors: These are inaccuracies observed at the element level within the generated table. Element errors encompass incorrect numbers, values, or inappropriately empty cells, reflecting discrepancies in individual table entries relative to the correct table. In figure 6 and figure 7, most errors are element errors.

A.3 Metric Effectiveness

Table 4 crystallizes different results of examples in Appendix A.1 based on our H-score metric. Taking GPT-4 as an example, there exist slight drops from 2.0 to 1.86 in content H-score, and very accurately, this trend is followed by the results of GPT-4 revealing slight errors in its content. For the format H-score, GPT-4 also cannot do very well for its numeric performance, which is not close enough to the full score, matched by the poor performance of GPT-4 in this area. Other models’ H-scores follow this trend as well. Therefore, the H-score can respond to differences in both content and format accurately.

Model	Content H-score	Format H-score
GPT-3.5	1.39	4.0
GPT-4	1.86	3.44
Vicuna-13B	1.51	4.0
LLaMA2-7B	1.37	4.0
Ours-7B	2.0	4.0
Reference	2.0	4.0

Table 4: H-scores for different results of Ours-7B, GPT-3.5, GPT-4, Vicuna-13B and LLaMA2-7B.

B MTurk

To facilitate a comprehensive analysis of the LLM output, we designed a task on Amazon Mechanical Turk (MTurk) to gather detailed annotations about various error types encountered in the generated structured tables. The task was structured as follows:

From the RotoWire dataset, we randomly chose 100 instances of LLM-generated output, ensuring a representative mix of quality based on preliminary assessment.

Each Human Intelligence Task (HIT) presented the annotators with a side-by-side view of the LLM output and the expected structured table format. Annotators were instructed to identify and categorize errors according to predefined types: ‘Element Errors’, ‘Element Format Errors’, ‘Structure Error’, and ‘Structure Naming Errors’.

We provided extensive guidelines, exemplified with step-by-step instructions, to clarify typical instances of each error type. These guidelines were reviewed and iteratively improved through a pilot study conducted with a small set of annotators.

About the qualifications of Amazon Mechanical Turk (MTurk) workers, we use the following qualifications to recruit in total of 10 MTurk workers with good track records: HIT approval rate greater than or equal to 98%, number of HITs approved greater than or equal to 500, and located in one of the following English native-speaking countries: Australia, Canada, New Zealand, United Kingdom, United States. Each annotator is limited to annotating 10 examples, including both the output of GPT-3.5 and GPT-4.

Annotators workers were compensated \$7, calibrated to equal a \$42/hour pay rate. We first annotated examples in-house to determine the required annotation speed. A summary block usually takes around 10 minutes.

To demonstrate our annotation template and facilitate future research, we show the interface for annotations.

Your Task

Sign in to Google to save your progress. [Learn more](#)

Instruction
 In this task, you will be presented with a text description of a table, along with three different table representations: two generated by AI and one correct table. Your task is to evaluate and score the AI-generated tables based on their accuracy and formatting, using a scale of 0 to 1, where 0 means completely inaccurate/poorly formatted and 1 means completely accurate/well-formatted.

Additionally, if there are any errors in the AI-generated tables, you are required to identify whether they are format errors or content errors. Here's a brief description of these two types of errors:

Format Errors: These occur when the structure, layout, or arrangement of the table does not adhere to standard table conventions or does not match the format indicated in the text description.

Content Errors: These occur when the information, data, or details in the table do not align with the information given in the text description.

After identifying the type of error, you should categorize it according to the following categories: Element Errors, Format Errors of Elements, Empty Element Issues, Excess or Missing Rows/Columns, Row or Column Naming Errors.

Remember, your evaluations are crucial in helping us understand and improve the abilities of AI in generating structured data. Thank you for your contributions!

Text
 The Grizzlies (50) used a strong second half to outlast the Suns (3 - 2) 102 - 91 in Phoenix on Wednesday night. Memphis found itself behind six at halftime but outscored Phoenix 30 - 19 in the third quarter and 26 - 20 in the final period. The Grizzlies shot 50 percent from the field, led by strong performances from Courtney Lee and Mike Conley. Lee scored 22 points (9 - 14 FG, 4 - 5 3Pt), while Conley led all scorers with 24 (9 - 14 FG, 3 - 4 3Pt) and 11 assists. Marc Gasol added 18 points, six assists, and five rebounds. The Suns, who beat the Lakers 112 - 106 on Tuesday, were paced by 23 points (9 - 12 FG), five rebounds and four assists from Eric Bledsoe. It was a quiet night for Goran Dragic, who scored just six points in 26 minutes. The third member of the backcourt trio, Isaiah Thomas, had 15 points and two assists off the bench, while Markieff Morris added 20 points and five rebounds. The Grizzlies out-rebounded Phoenix 37 - 35 and outscored the Suns in the paint 46 - 32. Memphis also registered 25 assists compared to only 13 - on 32 field goals - for the Suns. Memphis now heads to Oklahoma City to take on the Thunder on Friday. Phoenix, meanwhile, hosts the Kings on Friday. Table format: There are 2 tables. The first table has 10 columns and 2 rows. The column names are "Team", "Number of team assists", "Percentage of field goals", "Losses", "Total points", "Points in 3rd quarter", "Points

Figure 9: Our interface of Mturk.

Reference table
 Answer: Team: <NEWLINE> | Number of team assists | Percentage of field goals | Losses | Total points | Points in 3rd quarter | Points in 4th quarter | Rebounds | Wins | <NEWLINE> | Suns | 13 | 2 | 91 | 19 | 20 | 35 | 3 | <NEWLINE> | Grizzlies | 25 | 50 | 1 | 102 | 30 | 26 | 37 | | <NEWLINE> Player: <NEWLINE> | Assists | 3-pointers attempted | 3-pointers made | Field goals attempted | Field goals made | Minutes played | Points | Total rebounds | <NEWLINE> | Marc Gasol | 6 | 1 | 1 | 18 | 5 | <NEWLINE> | Courtney Lee | 15 | 4 | 14 | 9 | 22 | | <NEWLINE> | Mike Conley | 11 | 4 | 3 | 14 | 9 | 12 | 4 | <NEWLINE> | Markieff Morris | 1 | 1 | 1 | 20 | 5 | <NEWLINE> | Goran Dragic | 1 | 1 | 26 | 6 | | <NEWLINE> | Eric Bledsoe | 4 | 1 | 12 | 9 | 23 | 5 | <NEWLINE> | Isaiah Thomas | 2 | 1 | 1 | 15 | |

AI generated table A
 | Team | Number of team assists | Percentage of field goals | Losses | Total points | Points in 3rd quarter | Points in 4th quarter | Rebounds | Wins | | Suns | 13 | 47.7% | 2 | 91 | 19 | 20 | 35 | 0 | | Grizzlies | 25 | 50.0% | 1 | 102 | 30 | 26 | 37 | 1 | Table 2: | Player | Assists | 3-pointers attempted | 3-pointers made | Field goals attempted | Field goals made | Minutes played | Points | Total rebounds | | Marc Gasol | 6 | 0 | 12 | 8 | 35:00 | 18 | 5 | | Courtney Lee | 15 | 4 | 14 | 9 | 34:00 | 22 | 2 | | Mike Conley | 11 | 4 | 3 | 14 | 9 | 36:00 | 24 | 0 | | Markieff Morris | 1 | 1 | 5 | 2 | 14 | 8 | 34:00 | 20 | 5 | | Goran Dragic | 4 | 2 | 1 | 12 | 9 | 26:00 | 6 | 5 | | Eric Bledsoe | 4 | 3 | 1 | 12 | 9 | 34:00 | 23 | 5 | | Isaiah Thomas | 2 | 5 | 2 | 11 | 5 | 26:00 | 15 | 0 |

Score

	1	2	3	4	5	6	7	8	9	10	
Bad	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Good

Whether there are Element Errors

Yes

No

Figure 10: Our Interface of Mturk.

C Scoring

C.1 P-Score

Our approach involves prompting the model to engage in Chain-of-Thought reasoning before issuing its scores. Firstly, we instruct GPT on how to evaluate both "content similarity" and "structure similarity". Following this, the model is guided on the correct procedure to output its answer. To calculate the scores, the model is queried with both the predicted table and the ground truth table in varying sequences, after which the scores are averaged. We'll illustrate this process using the P-Scores prompt for raw text tables as an illustrative example:

"Based on the above, we wanted to determine if the above tables are similar. Ideally, they should have identical content and structure. Score the "content similarity" and "structural similarity" between 0 and 10.

- Content similarity: 10 if the contents of the table cells are identical, 0 if they are entirely different. If about 50% of the cells have the same data, the score should be 5.

- Structural similarity: 10 if the tables have the same structure (e.g. same column and rows with identical ordering, same alignment, etc.) although text formatting differences can be ignored (e.g. colors, font).

Output a JSON object such as the following:

```
"""json
{{
  "content_similarity": ...
  "structural_similarity": ...
}}
```

Think carefully, and then output the scores."

For instance, in figure 11, both tables have identical structures, so their structural similarity score is 10. The contents of the first table of Table1 and Table2 are the same, and the second table of Table1 and Table2 are almost 10% similar. Therefore their content similarity score is 5.

C.2 H-Score

We attach our algorithm to calculate H-Score as Algorithm 1.

L^AT_EX We use the `pylATEX` library to parse a given L^AT_EX table, and walk through the parse-tree structure in the `tabular` environment to identify the table "cells". We score the content similarity

based on strings within the cells, and score structural similarity based on having the matching number of rows and columns, the same caption, and the same cell alignment.

HTML We use the `beautifulsoup4` library to parse a given L^AT_EX HTML snippet and walk through the parse-tree structure in `<table>`, `` or `` tags to identify data cells. We separately build a tree of white-listed HTML tags to score the structural similarity, traversing an HTML document tree structure, disregarding the actual content within the tags and simplifying it by focusing only on specific HTML tags (defined in `RECOGNIZED_HTML_TAGS`). We score the content similarity based on strings within the cells and score structural similarity based on the similarity of the structure tree and the total number of cells matching.

White-listed HTML tags:

```
RECOGNIZED_HTML_TAGS = [
  "table", "tr", "th", "td",
  "ul", "ol", "li",
  "div", "span", "p",
  "a", "img", "embed", "pre",
  "h1", "h2", "h3", "h4", "h5", "h6",
  "input", "button",
```

]

Raw Text Tables In our evaluated dataset, each example consists of two tables (Team and Player). We do a string search for "Team" and "Player" headers to identify the two tables. We then parse the tables according to Markdown formatting, with newlines and pipes as row and column dividers respectively, to identify the table cells. We score the content similarity based on strings within the cells, and score structural similarity based on the similarity of column names and the number of rows and columns matching.

String Similarity Measurement: Our script includes methods to calculate the similarity between two strings. These methods can be used to compare the structure or content of HTML, latex documents, or any other pair of strings. The similarity is evaluated using well-established algorithms in text analysis: the Levenshtein distance and the `SequenceMatcher` from Python's `diff` module.

Table 1

Team	Percentage of Field Goals	Losses	Total Points	Points in 3rd Quarter	Wins
Bucks	54%	12	95	46	4
Cavaliers	39%	41	88	21	19

Player	Assists	Defensive Rebounds	3-Pointers Attempted	3-Pointers Made	Offensive Rebounds	Personal Fouls	Points	Total Rebounds	Steals
LeBron James	10	4	22	4	4	4	22	13	13
Kevin Love	13	13	13	13	4	4	13	13	13
Kyrie Irving	4	4	20	4	4	4	4	20	4
Jabari Parker	5	5	18	5	5	5	5	18	5
Michael Beasley	5	5	17	5	5	5	5	17	5

Table 2

Team	Percentage of Field Goals	Losses	Total Points	Points in 3rd Quarter	Wins
Bucks	54%	12	95	46	4
Cavaliers	39%	41	88	21	19

Player	Assists	Defensive rebounds	3-pointers attempted	3-pointers made	Offensive rebounds	Personal fouls	Points	Total rebounds	Steals
LeBron James	4						22	4	
Kevin Love							13	13	
Kyrie Irving							20		
Jabari Parker	1			1	1	1	18		
Michael Beasley		1	1	1	1		17		1

Based on the above, we wanted to determine if the above tables are similar. Ideally, they should have identical content and structure. Score the "content similarity" and "structural similarity" between 0 and 10.

- Content similarity: 10 if the contents of the table cells are identical, 0 if they are entirely different. If about 50% of the cells have the same data, the score should be 5.

- Structural similarity: 10 if the tables have the same structure (e.g. same column and rows with identical ordering, same alignment, etc.) although text formatting differences can be ignored (e.g. colors, font). Output a JSON object such as the following:

```
"""json
{
  "content_similarity": ...
  "structural_similarity": ...
}
"""
```

Think carefully, and then output the scores.

ChatGPT

To calculate the content and structural similarity between the two tables, we need to compare both the content and structure of the tables.

For content similarity:

- We'll compare each cell of the two tables and count how many cells have identical data.
- We'll then calculate the percentage of identical cells and score it accordingly.

For structural similarity:

- We'll compare the structure of the two tables, including the number of rows and columns, the order of rows and columns, and the presence of headers.
- We'll then score the similarity based on these factors.

Let's proceed with the calculation:

```
json
{
  "content_similarity": 5,
  "structural_similarity": 10
}
```

Explanation:

- Content similarity: The content similarity score is 5 because about 50% of the cells have the same data between the two tables.
- Structural similarity: The structural similarity score is 10 because both tables have identical structures with the same number of rows and columns, and the ordering of rows and columns is consistent.

Figure 11: An example of P-score calculation. The top two figures displays two tables, accompanied by a prompt below to calculate the content and structural similarity between them. The goal is to determine if the tables have identical content and structure. The content similarity will be scored based on the percentage of cells with identical data, ranging from 0 (no similarity) to 10 (complete similarity). The structural similarity will be scored 10 if the tables share the same structure, including columns, rows, and alignment. The output will be a JSON object with the calculated scores.

Algorithm 1: The provided algorithm is the H-score algorithm with pseudocode. It aims to calculate the content similarity (Content H-score) and format similarity (Format H-score) between predictions and references. The algorithm begins by defining functions for Levenshtein similarity and Difflib similarity to compute string similarities. It then iterates over each pair of predictions and references, parsing their structures and data to determine the similarity in terms of the number of columns, rows, column names, and data rows. The final step involves averaging the similarity scores to obtain the Content H-score and Format H-score.

Data: predictions α , references β

Result: Content H-score γ , Format H-score δ

```

1 Function LevenshteinSimilarity( $p, q$ ):
2   if  $q$  is empty then
3     |   return 0.0
4   end
5    $s \leftarrow 1 - \text{LevenshteinDistance}(p, q) / (2 * \text{length}(p))$ 
6   return  $s$ 
7 Function DifflibSimilarity( $p, q$ ):
8    $s \leftarrow \text{difflib.SequenceMatcher}(None, p, q)$ 
9   return  $s$ 
10  $\gamma \leftarrow 0$ 
11  $\delta \leftarrow 0$ 
12 for  $\sigma, \phi$  in zip( $\alpha, \beta$ ) do
13   Parse  $\sigma$  to get its number of columns  $\lambda_c$ , number of rows  $\lambda_r$ , column names  $\mu$  and data rows  $\kappa$ 
14   Parse  $\phi$  to get its number of columns  $\theta_c$ , number of rows  $\theta_r$ , column names  $\tau$  and data rows  $\omega$ 
15    $\rho_1 \leftarrow$  if  $\lambda_c == \theta_c$  then 1.0 else 0.0
    // Whether their number of columns are same
16    $\rho_2 \leftarrow$  if  $\lambda_r == \theta_r$  then 1.0 else 0.0
    // Whether their number of rows are same
17    $\rho_3 \leftarrow \text{LevenshteinSimilarity}(\mu, \tau)$ 
    // Compute their columns Levenshtein Similarity
18    $\rho_4 \leftarrow \text{DifflibSimilarity}(\mu, \tau)$ 
    // Compute their columns Difflib Similarity
19    $\rho_5 \leftarrow \text{LevenshteinSimilarity}(\kappa, \omega)$ 
    // Compute their data Levenshtein Similarity
20    $\rho_6 \leftarrow \text{DifflibSimilarity}(\kappa, \omega)$ 
    // Compute their data Difflib Similarity
21    $\gamma \leftarrow \gamma + \text{average}(\rho_5, \rho_6)$ 
22    $\delta \leftarrow \delta + \text{average}(\rho_1, \rho_2, \rho_3, \rho_4)$ 
23 end
24  $\gamma \leftarrow \gamma / \text{length}(\alpha)$ 
25  $\delta \leftarrow \delta / \text{length}(\alpha)$ 

```

D Prompt for FormatCoT and Inference

D.1 Prompt for FormatCoT

Raw Text Table Description Traditional data-to-text datasets only have raw text for each table. However, it is not enough for GPT-3.5 or other LLMs to generate correct tables. As a result, we added some format descriptions to help them generate the correct tables. We use GPT-3.5 to achieve this. We want to get detailed format information without concrete contents in cells, so we explicitly include these requirements in the prompt. Here is our prompt: “Describe details about the given text. First, give the number of tables, and then for each table, describe its format such as the number of columns and rows, column names, and row names.”

HTML Table Description Unlike data-to-text datasets, HTML datasets only have final outputs, so we are required to generate a detailed description of their format and content. For content descriptions, we can simply ask GPT-3.5 to output raw text without HTML tags. For format descriptions, however, we need to ask GPT-3.5 to describe each tag, otherwise, it will leave out some tags and describe the table in general rather than detailed information. Moreover, it is necessary to ask it to use specific numbers instead of ‘several’ or ‘multiple’. Here is our prompt for HTML format descriptions: “Describe the format of this HTML in detail according to each HTML tag of the following HTML code. Be careful and make sure don’t miss any HTML tags. Please use more than 300 words to explain the format. Use specific numbers rather than being vague about several.”

LaTeX Table Description Similar to HTML prompt generation, it is necessary to ask GPT-3.5 to generate both format descriptions and content descriptions as latex datasets only have final outputs. For content descriptions, we can simply ask GPT-3.5 to describe the given latex table as detailed as it can and include all cells. For format description, since the latex format is too complex, we need to give it a small example to learn. Then we ask GPT-3.5 to describe the detailed format of a given latex table, including specific questions to help it generate format descriptions. Here is our prompt for latex format descriptions: “Describe the detailed format of a given latex table according to the commands and tags with more than 500 words. Include: Whether there is table border lines? How is text

alignment? What are table attributes? Whether to bold? Whether to add \ref? Please clearly explain whether there are horizontal and vertical lines bordering each row and column. Say anything about a special "\ format token in latex if there is one. Don’t display latex code directly. Use natural language. And provide enough format information for me to recreate this table based on your output description.”

D.2 Prompt for Inference

When inferencing raw text tables, LLMs tend to output tabular results rather than raw text tables. As a result, we need to give it an example output first, then tell the model that the input consists of two parts, text and format descriptions, and ask the model to generate the output based on them. For HTML and Latex inference, we can simply ask models to infer from the input and specify the format and content sections in the input, since models can generate correct syntax.

FORMATCOT Prompt for Raw Text Table	Describe details about the given text. First, give the number of tables, and then for each table, describe its format such as the number of columns and rows, column names, and row names.
FORMATCOT Prompt for HTML Table	Describe the format of this HTML in detail according to each HTML tag of the following HTML code. Be careful and make sure don't miss any HTML tags. Please use more than 300 words to explain the format. Use specific numbers rather than being vague about several.
FORMATCOT Prompt for LaTeX Table	Describe the detailed format of a given latex table according to the commands and tags with more than 500 words. Include: Whether there are table border lines? How is text alignment? What are table attributes? Whether to bold? Whether to add \ref? Please clearly explain whether there are horizontal and vertical lines bordering each row and column. Say anything about a special "" format token in latex if there is one. Don't display latex code directly. Use natural language. And provide enough format information for me to recreate this table based on your output description.
Prompt for Inference	Based on the example output above, generate the raw text/HTML/LaTeX table according to the following description.

Table 5: Our prompts for FORMATCOT and Inference. The prompt requests an overview of table formatting in raw text, HTML, and LaTeX formats, including descriptions of the number of tables, column and row structures, formatting elements, and specific instructions. The prompt for inference illustrates the request to generate the tables based on the provided details.

E Ability Map

Based on our automated evaluation, we selected Vicuna, GPT-3.5, GPT-4, and Ours as representative models and conducted an in-depth analysis of the causes of model errors.

We identified content accuracy, formatting, numerical reasoning, and handling of long tables as the main sources of these errors.

At the fundamental level, we decompose the process of model-generated complex structured outputs into two parts: Content Selection and Format Planning. Initially, the model needs to identify key information from a given vast amount of unstructured input, extract this information, understand it, and organize it. Subsequently, it needs to plan how to summarize these extracted details, devise the format of the table to be generated, and then fill in the information.

Accordingly, we can break down the model’s capabilities into Coverage, Formatting Reasoning, Comprehension, Pragmatics, and Hallucination Control.

Coverage entails the model’s ability to accurately cover the content in the input. Formatting Reasoning pertains to judgment about the output format, assessing if the model can find the most appropriate and reasonable structured format.

Comprehension reflects whether the model can understand the content of the input, as there are times when it is necessary to infer from a large amount of data (including performing addition or subtraction or comparing multiple elements).

Pragmatics involves the ability to utilize special formats, such as HTML tags and specific syntax in LaTeX.

Finally, Hallucination Control signifies the model’s ability to refrain from generating content not present in the input.

We carried out manual annotations and obtained visualized results to demonstrate these aspects.

F Ablation Study for FormatCoT

F.1 Contrast between descriptions

In this section, we conduct an ablation study to examine the impact of our proposed FORMATCOT. In the generation of table descriptions sans FORMATCOT, we simply utilize the prompt: “Provide a description of the following tables.” We display the results in figure 12. The primary differentiation between results pivots on the extent of details incorporated.

For instance, in the FORMATCOT result, the description comprises an array of detailed format information - encompassing row names, column names, and table count. The precision in these details proves substantial enough for models to accurately recreate the tables in question.

Contrastingly, the outcome bereft of FORMATCOT conveys considerably less information - providing incomplete column names without the accompaniment of row names. This sparse degree of detail proves insufficient for models seeking to faithfully regenerate the corresponding tables.

F.2 Contrast between results

In this section, we draw a comparison between two sets of description results, shown in figure 13. The FORMATCOT result showcases a table that stands remarkably close to the correct table, albeit with minor errors. It contains an extra row termed “Player” in the initial table, a discrepancy potentially attributable to the fact that the result comprises two tables, with “Player” denoting the header of the subsequent table. We posit that this error could potentially be circumvented with a different method of integrating table names.

Furthermore, an additional column surfaces in the second table, which in reality represents the final row of that table. Besides these minor inaccuracies, the FORMATCOT result accurately replicates the content in each cell as well as maintaining the overall format.

Conversely, the alternative result contains multiple errors that span both content and format. Initially, an additional row is present in the first table, introducing an unrelated basketball team that bears no relevance to the game under consideration. Following this, the second table possesses an excessive number of player names, encompassing unnecessary players along with coaches who did not participate in the game.

Furthermore, its content is not entirely accurate, with discrepancies present in the statistics attributed to both Gordon Hayward and Gerald Green. These shortcomings underscore the efficiency and essentiality of implementing the FORMATCOT to ensure accuracy and precision.

Target tables

Team	Losses	Total Points	Wins
Suns	38	87	39
Jazz	42	85	34

Player	Assists	Blocks	3-Pointers Made	Field Goals Attempted	Minutes Played	Points	Total Rebounds	Steals
Gordon Hayward	-	-	-	18	-	21	-	-
Rudy Gobert	2	1	-	-	38	8	15	1
Rodney Hood	-	-	-	-	-	17	-	-
Brandan Wright	1	4	-	-	33	14	3	3
TJ Warren	-	-	-	-	-	4	-	-
Gerald Green	-	-	3	-	22	24	-	-

FormatCoT

There are two tables. The first table has four columns and three rows. The column names are "Team", "Losses", "Total points", and "Wins". The row names are "Suns", "Jazz", and "Player". The second table has nine columns and six rows. The column names are "Assists", "Blocks", "3-pointers made", "Field goals attempted", "Minutes played", "Points", "Total rebounds", "Steals", and "Gerald Green". The row names are "Gordon Hayward", "Rudy Gobert", "Rodney Hood", "Brandan Wright", "TJ Warren", and "Gerald Green".

Without FormatCoT

The first table lists basketball teams with their number of losses, total points, and wins. The second table lists basketball players with their number of assists, blocks, 3-pointers made, field goals attempted, minutes played, points, total rebounds, and steals. Some cells within the second table are empty, meaning that data is not available for those specific categories for certain players.

Figure 12: Using FORMATCOT and normal instructions to ask GPT-3.5 to generate table descriptions based on the input text, FORMATCOT results contain more detailed information about row names.

FormatCoT

Team	Losses	Total Points	Wins
Suns	38	87	39
Jazz	42	85	34
Player			

Player	Assists	Blocks	3-Pointers Made	Field Goals Attempted	Minutes Played	Points	Total Rebounds	Steals	Gerald Greens
Gordon Hayward	-	-	-	-	-	21	-	-	
Rudy Gobert	2	1	-	-	38	8	15	1	
Rodney Hood	-	-	1	-	-	17	-	-	
Brandan Wright	1	4	-	-	33	14	3	3	
TJ Warren	-	-	-	-	-	4	-	-	
Gerald Green	-	-	3	-	22	24	-	-	

Without FormatCoT

Team	Losses	Total Points	Wins
Phoenix Suns	38	87	39
Utah Jazz	42	85	34
Oklahoma City Thunder			

Player	Assists	Blocks	3-Pointers Made	Field Goals Attempted	Minutes Played	Points	Total Rebounds	Steals
Rodney Hood	-	-	-	-	-	-	-	-
Gerald Green	-	-	-	-	-	24	-	-
Brandan Wright	1	4	-	-	33	14	3	3
Trey Burke	-	-	-	-	-	-	-	-
T.J. Warren	-	-	-	-	-	4	-	-
Dante Exum	-	-	-	-	-	-	-	-
Joe Ingles	-	-	-	-	-	-	-	-
Gordon Hayward	2	1		18		21	15	1
Rudy Gobert	2	1			38	8	15	1

Figure 13: Using two descriptions to regenerate table descriptions based on the input text and descriptions, FORMATCoT result is more correct in both format and content.