Text-Tuple-Table: Towards Information Integration in Text-to-Table Generation via Global Tuple Extraction

Anonymous ACL submission

Abstract

The task of condensing large chunks of textual information into concise and structured tables has gained attention recently due to the emergence of Large Language Models (LLMs) and their potential benefit for downstream tasks, such as text summarization and text mining. Previous approaches often generate tables that directly replicate information from the text, limiting their applicability in broader contexts, as text-to-table generation in real-life scenarios necessitates information extraction, reasoning, and integration. However, there is a lack of both datasets and methodologies towards this task. In this paper, we introduce LIVESUM, a new benchmark dataset created for generating summary tables of competitions based on real-time commentary texts. We evaluate the performances of state-of-the-art LLMs on this task in both fine-tuning and zero-shot settings, and additionally propose a novel pipeline called T^3(**T**ext-**T**uple-**T**able) to improve their performances. Extensive experimental results demonstrate that LLMs still struggle with this task even after fine-tuning, while our approach can offer substantial performance gains without explicit training. Further analyses demonstrate that our method exhibits strong generalization abilities, surpassing previous approaches on several other text-to-table datasets.

1 Introduction

Reading extensive texts is demanding and time-consuming for humans, further compounded by the challenge of effectively capturing the key elements. Consequently, recent works have shifted to explore the structured summarization of text (Jain et al., 2024), with tables being one highly prevalent form (Wu et al., 2022; Li et al., 2023c; Sundar et al., 2024). These approaches improve text comprehension by extracting inherent yet valuable structural information from long unstructured text and enabling their applications in downstream scenarios,

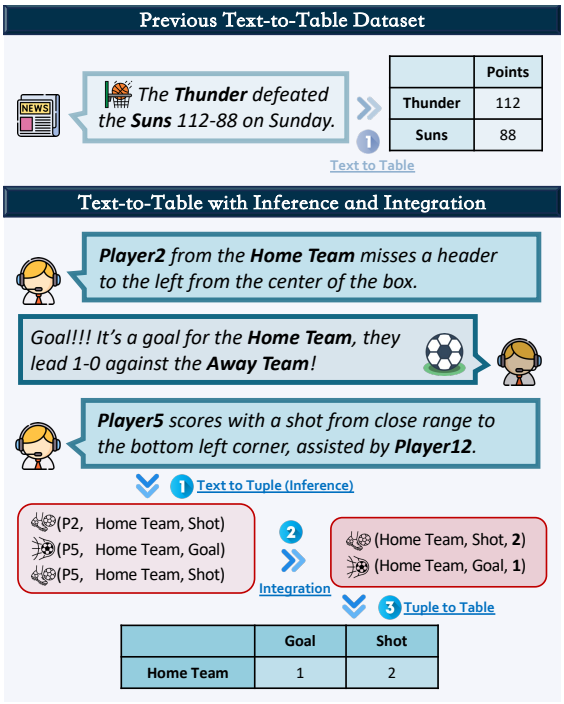

Figure 1: An overview of the differences between our proposed LIVESUM dataset and previous dataset (Wiseman et al., 2017), as well as our proposed pipeline called T^3(**T**ext-**T**uple-**T**able) which consists of three steps.

such as question answering (Chen et al., 2020; Zhu et al., 2024), text summarization (Wiseman et al., 2017; Wang et al., 2020; Mulwad et al., 2023) and text data mining (Li et al., 2023b; Sui et al., 2024).

However, previous studies on text-to-table generation primarily rely on datasets traditionally used for table-to-text tasks (Wiseman et al., 2017; Novikova et al., 2017). One evident issue is that these tasks focus merely on format transformation, where the information in the table and the corresponding text representation are essentially similar (Lebret et al., 2016; Bao et al., 2018). For example, in the upper part of Figure 1, the table can be easily completed by extracting relevant numbers from the text without intermediate inference. Such seemingly meaningful correlations can introduce bias into the models, causing them to excel at

replicating relevant information but struggle when it comes to categorizing and integrating numbers in complex scenarios. This is further evidenced by the fact that fine-tuning models already perform very well and greatly surpass zero-shot LLMs (Tang et al., 2023; Sundar et al., 2024). Hence, a more complicated dataset that requires information aggregation and minimizes the presence of spurious correlations, closely resembling real-world scenarios, is definitely needed for a more rigorous evaluation of models' text-to-table generation proficiency.

Apart from the research gap at the benchmark level, in terms of methodology, considerable attention has been given to studying the ability of LLMs to comprehend and generate complex structured outputs (Tang et al., 2023; Jain et al., 2024), driven by the exceptional success of LLMs in recent years (Touvron et al., 2023; Anthropic, 2024; OpenAI, 2024). Extensive benchmarks indicate that LLMs exhibit sub-optimal performance in zero-shot settings, with multiple cases of generating inaccurate contents deviate from the given text (Tang et al., 2023). To address this issue, more sophisticated prompting mechanisms have been proposed (Wei et al., 2022; Khot et al., 2022; Dua et al., 2022). Among them, Jain et al. (2024) introduce a divide-and-generate prompting approach to generate more accurate and informative tables, demonstrating its effectiveness in improving model performance. However, this simplistic approach of dividing text into paragraphs and generating tables is unsuitable in more complicated situations because table-relevant information may not be contiguous in the original text and may span across various paragraphs. Therefore, developing a robust prompting method is also needed for generating complex tables that capture crucial information from scattered text or paragraphs.

To resolve the aforementioned research gaps, we introduce a novel benchmark, LIVESUM, which consists of 3,771 text-based live commentaries from real-world football matches, intending to evaluate the models' ability to generate summary tables. Unlike previous benchmarks, our benchmark necessitates the model to possess the ability to extract correct and meaningful information from complex textual data, specifically emphasizing information integration, reasoning, and conceptualization skills (Wang et al., 2023a). This is because commentaries in close temporal proximity or with similar semantic meanings may describe the same event, while verbs with similar meanings may refer to the same types of events. For example, in Figure 1, the second and third dialogue boxes both describe the same goal event, and the verbs "*goal*" and "*score*" refer to the same goal event.

Along with the benchmark, we also introduce a robust prompting-based method T^3 to address our proposed task. Specifically, our method draws inspiration from the inherent attributes of the table, where each cell, along with its corresponding row header and column header, creates an informative triple (namely `(row header, column header, cell)`), which degenerates into a binary tuple when lacking row or column headers. These tuples serve as cues for humans to locate specific information in the text and complete the table accordingly. Consequently, our pipeline begins by extracting the relevant tuples from the text, followed by the integration of these tuples, and ultimately generating one or more summary tables.

We hope that the proposed dataset, method, and experimental results can provide valuable insights for tasks such as text-to-table, as well as any task involving the generation of complex structured outputs from text. In summary, in this paper, we make the following contributions:

- To the best of our knowledge, LIVESUM is the first benchmark dataset designed to evaluate the information integration ability of models in text-to-table generation tasks.
- We introduce a novel T^3(**T**ext-**T**uple-**T**able) prompting pipeline that functions as a flexible framework, applicable to any text-to-table generation tasks.
- We conduct extensive experiments to evaluate the performance of LLMs under different settings and demonstrate that our T^3 pipeline can bring significant improvements while showcasing excellent generalization capabilities.

2 Task Definition

We first provide a formal definition of the text-to-table generation. The input $\mathcal{S}$ consists of a textual passage with n tokens, denoted as $\mathbf{x} = x_1, \ldots, x_n$, and optionally, an instruction text with m tokens, denoted as $\mathbf{y} = y_1, \ldots, y_m$, which provides guidance on the format or content of the generated tables. The output $\mathcal{T}$ is a set of $k(k \geq 1)$ tables, $\mathbf{T}^1, \ldots, \mathbf{T}^k$. For the output tables, we present a more detailed definition that covers two aspects: structure-related and content-related.

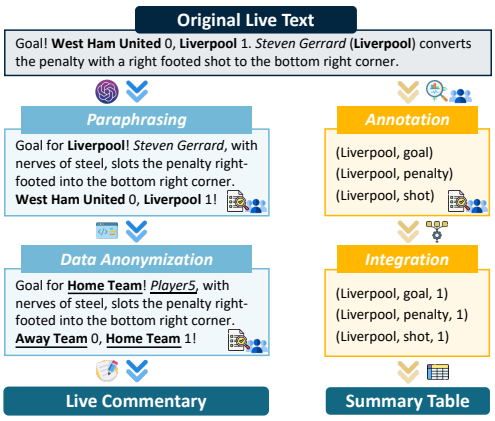

Figure 2: Overview of the pipeline for constructing the LIVESUM dataset illustrated with a sample sentence.

Structure We assume there are no merged cells in the tables for simplicity. Each table $\mathbf{T}^i$ has a caption $\mathbf{c}^i = c^i_1, \ldots, c^i_d$, where $d = |\mathbf{c}^i|$, and it consists of n^i_r rows and n^i_c columns, resulting in a total of $n^i_r \times n^i_c$ cells. The cell $\mathbf{T}^i_{p,q}$ in the p-th row and q-th column is composed of a sequence of tokens: $\mathbf{T}^i_{p,q,1}, \ldots, \mathbf{T}^i_{p,q,r}$, where $r = |\mathbf{T}^i_{p,q}|$. The table $\mathbf{T}^i$ must have either a row header for all rows or a column header for all columns and it is also possible for the table to have both.

Content We define that the information in the output tables should be derived from the input $\mathbf{x}$ or can be inferred from $\mathbf{x}$. For each cell $\mathbf{T}^i_{p,q}$ in table $\mathbf{T}^i$, when combined with its row header $\mathbf{T}^i_{p,1}$ (if any), its column header $\mathbf{T}^i_{1,q}$ (if any), and the table's caption $\mathbf{c}^i$, it should convey the equivalent information as expressed in the input $\mathbf{x}$ and complies with the instruction $\mathbf{y}$ (if any).

3 LIVESUM Dataset

We consider the problem of generating match statistic information tables from textual live commentary. Inspired by ROTOWIRE (Wiseman et al., 2017), a data-to-document dataset in the sports domain that aims to generate textual summaries by incorporating statistical data from basketball games, we instead focus on live commentary in football, which is available on BBC Sports[1]. We crawl the data for the English Premier League from 2014 to 2023 and obtain complete commentary for 3,771 matches. Figure 2 shows the pipeline we use to construct the dataset. Section 3.1 describes the generation process of the live commentary, and Section 3.2 describes the generation process for the summary table. More details are provided in Appendix A.

[1] https://www.bbc.com/sport/football

Figure 3: Eight types of event information (inner circle) that require summarization in LIVESUM dataset, along with their common expressions (outer circle) in the commentary.

3.1 Live Commentary Generation

To address formatting issues in the original textual live commentary on the website, we paraphrase the text to match the commentator's style while ensuring a certain degree of diversity. Building upon previous studies (Kim et al., 2023a; Chen et al., 2023; Kim et al., 2023b), we employ ChatGPT (OpenAI, 2022) to generate complete live commentary automatically. Subsequently, to comply with privacy regulations and prevent bias in LLM benchmarks, we also anonymize the data leveraging named entity recognition (NER) techniques (Qi et al., 2020) to produce the final version of the live commentary.

3.2 Summary Table Generation

On the other hand, human annotators label a summary table for each match's commentary. We recruit five workers who are interested in football and are from English-speaking countries to perform the annotations. Since the occurrence of the events in football matches is deterministic, the ground truth is essentially unambiguous. In cases where there are inconsistencies in the annotated results, the correct answer is determined through a majority vote.

3.3 Statistics

LIVESUM comprises a collection of 3,771 pairs, consisting of textual live commentaries and corresponding summary tables. We randomly split the entire dataset into training and test sets, resulting in 3,017 instances for the training set and 754 instances for the test set. On average, each live commentary segment consists of 1,256 words. LIVESUM focuses on eight types of events, with the names and their corresponding common descriptions displayed in Figure 3.

4 T^3(Text-Tuple-Table) Pipeline

Our proposed T^3(Text-Tuple-Table) pipeline is designed to mimic the intuitive steps followed by humans when performing this task. When individuals aim to summarize a table from text, they typically extract pertinent or valuable tuples from the content, guided by any provided instructions, and then organize these tuples into one or more tables. Based on this concept, we divide this transformation into three stages: text-to-tuple, integration, and tuple-to-table, each of which is discussed in the following subsections. Taking Figure 1 as an example, we first extract key events mentioned in the text, then aggregate this information into consolidated tuples, and ultimately compile them into a table.

4.1 Text-to-Tuple

Considering the superior performance and flexibility of LLMs in information extraction compared to traditional techniques (Ma et al., 2023; Xu et al., 2023), we employ an LLM as our tuple extractor. We follow the instructions from InstructUIE (Wang et al., 2023b) and design the following prompting:

Text-to-Tuple Prompting Template

According to `<Instruction>`, please extract the relevant events and information in the form of tuples, structured as (*subject*, *object*, *verb*) or (*subject*, *attribute*, *value*): `<Text>`

where `<Instrction>` is the directive for the current task, and `<Text>` is the text to be transformed.

4.2 Information Integration

In this stage, we propose two approaches for integrating information. The first one involves direct execution by the LLM, using prompting to consolidate tuple data. The second one uses algorithms and code generated by the LLM to integrate tuple information, inspired by the LLMs' great success in code generation tasks (Roziere et al., 2023; Luo et al., 2023; Guo et al., 2024). T^3 defaults to using code generation in this step. The promptings for these two methods are shown as follows:

Information Integration Prompting Template

Direct Execution: According to `<Instruction>`, please integrate these tuples as required: `<Tuples>`

Code Generation: According to `<Instruction>`, please develop an algorithm to consolidate these tuples as specified: `<Tuples>`

where `<Instrction>` is the directive for the current task, and `<Tuples>` consists of the tuples extracted in the prior stage.

4.3 Tuple-to-Table

After obtaining the integrated tuples, we follow the previous implementation (Tang et al., 2023; Jain et al., 2024) and use the following prompting to generate the final tables:

Tuple-to-Table Prompting Template

According to `<Instruction>`, please generate one or more tables based on the following tuples: `<Tuples>`

where `<Instrction>` is the directive for the current task, and `<Tuples>` consists of the tuples produced in the prior stage.

5 Experimental Setup

Baseline Models In this study, we conduct fine-tuning on the LIVESUM dataset using three representative open-source LLMs: Mistral-7B-Instruct-v0.2 (Jiang et al., 2023), LLaMA-2 Chat 7B and LLaMA-2 Chat 13B (Touvron et al., 2023). We fine-tune these models following the current state-of-the-art fine-tuning methodologies (Tang et al., 2023). Therefore, the outcomes represent the best results achievable with the present fine-tuning methods. We also evaluate eight state-of-the-art LLMs in zero-shot settings: LLaMA-2 Chat 13B, LLaMA-2 Chat 70B (Touvron et al., 2023), Mistral Large (MistralAI, 2024), Claude 2.1 (Anthropic, 2023), Claude 3 Opus (Anthropic, 2024), ChatGPT (OpenAI, 2022), and GPT-4 (OpenAI, 2024). For each model, we conduct tests using two types of prompts. The first type directly describes the task by providing an instruction text $\mathbf{y}$ and accompanying it with the text $\mathbf{x}$. The second type uses the Chain-of-Thought (CoT) prompting (Wei et al., 2022), incorporating the phrase "let's think step by step" into the instruction text. See more details in Appendix C.

Evaluation Metric As the generated cell content in this task consists of numerical values, we utilize commonly employed metrics in regression tasks, namely the Root Mean Square Error (RMSE). We also report the Error Rate (ER) for each cell, defining a cell as erroneous if its content does not exactly match the ground truth.

Grouping by Event Difficulty Furthermore, we categorize the eight types of events into three

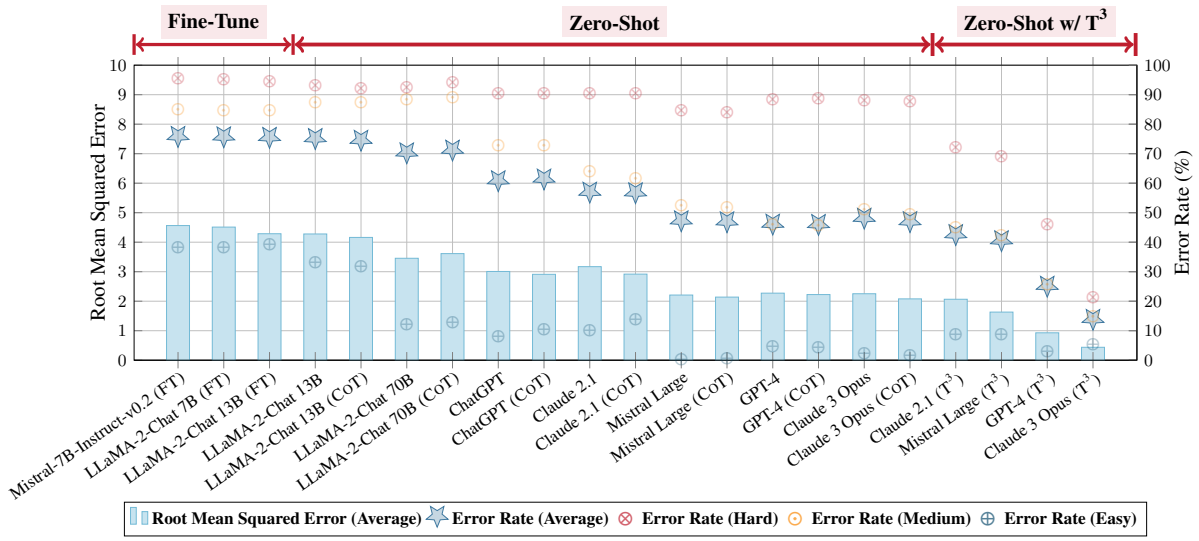

Figure 4: The performance of various LLMs under fine-tune and zero-shot settings, as well as after the application of the T^3 method on the test set of LIVESUM dataset. The average RMSE and error rate for each model are displayed, along with the error rate for each of the three difficulty sections. More results are in Table 1.

groups based on assessed difficulty: `Goals`, due to direct descriptions of scores in the original text, and `Red Cards`, due to their rare occurrence are categorized into the *Easy* section. `Shots` and `Fouls`, due to their varied expressions and descriptions, are classified into the *Hard* section. The remaining four event types are classified as *Medium* section. We report the RMSE and ER for each model across different difficulty categories to provide a more comprehensive analysis.

6 Experiments and Analyses

In this section, we will benchmark the performance of current state-of-the-art LLMs on the LIVESUM dataset, and further evaluate the effectiveness and generalization of our proposed approach. We aim to answer the following research questions:

RQ1 (Benchmarking) How do the current state-of-the-art LLMs perform on this dataset in fine-tuning and zero-shot settings?

RQ2 (Effectiveness) How does our proposed T^3 pipeline impact model performance?

RQ3 (Generalization) How effective is the T^3 pipeline when applied to other real-world datasets for the text-to-table generation task?

6.1 Benchmarking (RQ1)

We first analyze the performance of existing state-of-the-art LLMs on the LIVESUM dataset under fine-tuning and zero-shot settings, with results displayed in Figure 4 and Table 1. Overall, the performance of most models in the zero-shot setting far exceeds that in the fine-tuning setting, indicating that the previous state-of-the-art fine-tuning method has limited capability for information integration, and there is substantial room for improvement on this benchmark. In the zero-shot setting, it is noteworthy that most models show a slight improvement in both metrics after applying CoT. Among them, the best-performing models are Mistral Large, GPT-4, and Claude 3 Opus, which are nearly comparable. They achieve RMSEs ranging from 2.08 to 2.27 and error rates between 46.20% and 48.33%. Nevertheless, this still highlights a notable deficiency in the information integration capabilities of LLMs in the zero-shot settings, underscoring the challenges and significance of our benchmark. We then analyze performance across three categories of difficulty.

Easy Section It can be observed that the error rate of the fine-tuned models is generally around 40%, with an RMSE close to 1. In the zero-shot setting, LLaMA-2-Chat, ChatGPT, and Claude 2.1 models exhibit relatively poor performance, occasionally producing anomalously large values. The error rates of the other models generally remain below 5%, with RMSEs less than 0.2. Among these, the Mistral Large model performs the best, with both metrics significantly lower than other models.

Medium Section The medium section exhibits the greatest variation among models and serves as a crucial determinant of overall model performance. We organize the models based on their performance, with error rates in the zero-shot set-

Model	Easy		Medium		Hard		Average	
	RMSE	ER	RMSE	ER	RMSE	ER	RMSE	ER
♦ ***Fine-Tune***								
Mistral-7B-Instruct-v0.2	1.045	**38.36**	3.832	85.11	7.115	95.52	4.564	76.03
LLaMA-2-Chat 7B	1.047	38.41	3.728	84.91	7.107	95.40	4.512	75.91
LLaMA-2-Chat 13B	**1.043**	39.22	**3.587**	**84.60**	**6.671**	**94.42**	**4.287**	**75.71**
♦ ***Zero-Shot***								
LLaMA-2-Chat 13B	0.775	33.29	4.554	87.37	5.203	93.29	4.279	75.33
LLaMA-2-Chat 13B (CoT)	0.780	31.83	4.376	87.42	5.088	92.35	4.162	74.75
LLaMA-2-Chat 70B	0.410	12.34	3.189	88.59	4.941	92.41	3.455	70.48
LLaMA-2-Chat 70B (CoT)	0.450	12.86	3.221	89.25	5.314	94.24	3.613	71.40
ChatGPT	0.200	8.06	2.864	72.73	4.257	90.62	3.008	61.03
ChatGPT (CoT)	0.229	10.61	2.809	72.75	4.087	90.38	2.911	61.62
Claude 2.1	1.014	10.08	2.581	63.99	4.621	90.58	3.171	57.16
Claude 2.1 (CoT)	1.496	14.06	2.291	61.70	4.081	90.38	2.918	56.96
Mistral Large	**0.005**	**0.27**	2.385	52.45	2.712	84.62	2.209	47.45
Mistral Large (CoT)	0.018	0.73	2.311	51.82	**2.608**	**84.08**	2.139	47.12
GPT-4	0.156	4.64	**1.167**	46.05	4.114	88.53	2.273	46.32
GPT-4 (CoT)	0.154	4.38	1.173	**45.86**	3.981	88.73	2.225	**46.20**
Claude 3 Opus	0.078	2.52	1.617	51.36	3.713	88.06	2.253	48.33
Claude 3 Opus (CoT)	0.040	1.59	1.642	49.60	3.265	87.86	**2.079**	47.17
♦ ***Zero-Shot with*** $\mathbf{T^3}$								
Claude 2.1 (T^3)	0.193	8.95	1.965	44.99	2.751	72.15	2.066	42.77
Mistral Large (T^3)	0.191	8.82	1.596	42.37	2.136	69.23	1.631	40.70
GPT-4 (T^3)	**0.056**	**3.18**	0.854	25.83	1.219	46.22	0.929	25.27
Claude 3 Opus (T^3)	0.081	5.30	**0.406**	**14.79**	**0.477**	**21.29**	**0.438**	**14.04**

Table 1: The performance of various LLMs under three settings, showing RMSE and error rate across three difficulty categories and overall average. We **bold** the best results and underline the second-best results in each setting.

ting ranging from 89.25% down to 45.86%. GPT-4 with CoT performs the best, achieving the lowest error rate, which still indicates the suboptimal capabilities of LLMs.

Hard Section The charts clearly show that in the hard section, the zero-shot method shows minimal enhancement compared to fine-tuning, as most error rates are around 90%. However, Mistral Large is an exception, achieving a lower error rate of 84.08%, which demonstrates the challenging nature of the hard section.

6.2 Effectiveness (RQ2)

We apply the T^3 pipeline to four LLMs: Claude 2.1, Mistral Large, GPT-4, and Claude 3 Opus; the rest are not applicable to this method due to their ineffective extraction of tuples from inputs of such lengths, resulting in a minimal number of tuples or a substantial duplication of the same tuple. The implementation details are discussed in Appendix B. From Figure 4 and Table 1, it is observable that Claude 2.1 and Mistral Large exhibit similar improvements after applying the T^3 method, both showing slight enhancements over the best results in the zero-shot setting, with reductions in RMSE of 34.9% and 26.2% respectively and decreases in error rate of 25.2% and 14.2%. In contrast, GPT-4 and Claude 3 Opus display substantial improvements after implementing the T^3 method, with RMSE reductions of 59.1% and 80.6%, respectively, and error rate reductions of 45.4% and 70.9%. The reductions in these metrics are primarily reflected in the significant decrease in the error rate for the hard sections, creating a clear distinction from the zero-shot approaches. We then conduct an ablation study to evaluate the impact of the T^3 method on model performance. We experiment with two variant methods: T^3-MERGED (T^3M) and T^3-DIRECT-EXECUTION (T^3D). The former method involves using a single prompt that instructs the model to first extract relevant tuples before generating the table, while the latter one modifies the second step of the T^3 method to be directly executed by the LLM, rather than using code generation (see Appendix B.5 for more details). Table 2 presents the results using GPT-4 as an example. It is apparent that using T^3 considerably enhances the model's overall performance, leading significantly in overall metrics, with average reductions in RMSE and error rate of 59.1% and 45.4%, respectively. Compared to GPT-4 and CoT, the variants T^3M and T^3D also exhibit notable im-

Model	Easy RMSE/ER	Medium RMSE/ER	Hard RMSE/ER	Average RMSE/ER
GPT-4	0.16/4.6	1.17/46.1	4.11/88.5	2.27/46.3
w/ CoT	0.15/4.4	1.17/45.9	3.98/88.7	2.23/46.2
w/ T^3M	**0.00/0.1**	1.42/43.2	2.46/82.8	1.62/42.3
w/ T^3D	0.09/4.5	1.12/40.1	2.23/81.4	1.42/41.5
w/ T^3	0.06/3.2	**0.85/25.8**	**1.22/46.2**	**0.93/25.3**

Table 2: The ablation study results comparing the performance of different prompting methods. We **bold** the best results and underlined the second-best results.

Metric	Zero-Shot ChatGPT	Zero-Shot w/ T^3	Fine-Tune LLaMA-7B
SacreBLEU	77.58	78.91	**90.60**
ROUGE-L	86.11	88.36	**88.98**
BERTScore	96.75	97.34	**98.54**
BLEURT	64.66	**67.47**	66.07
BARTScore	-2.08	-1.90	**-0.69**
Content P-Score	6.84	7.29	**7.69**
Format P-Score	9.70	**9.88**	8.60
Content H-Score	1.66	**1.68**	1.65
Format H-Score	3.28	**3.63**	3.61

Table 3: The evaluation results on the test set of STRUC-BENCH Table dataset with nine metrics. We **bold** the best results and underlined the second-best results.

provements, with respective reductions in RMSE of 28.7% and 37.6%, and error rates of 8.6% and 10.3%. Full ablation studies are in Appendix D.

6.3 Generalization (RQ3)

To examine the generalization capabilities of T^3, we apply T^3 to two additional datasets designed to test text-to-table performance and compare it with previous methods. Section 6.3.1 involves table generation without the need for information integration, while Section 6.3.2 focuses on table generation without instructions.

6.3.1 Performance of T^3 on STRUC-BENCH Table Dataset

We test the performance of T^3 on the text-to-table benchmark STRUC-BENCH Table (Tang et al., 2023). This benchmark is based on the ROTOWIRE dataset (Wiseman et al., 2017) and employs traditional evaluation metrics, prompting score(*P-Score*), and heuristical score(*H-Score*), to conduct a comprehensive assessment of the output tables. They also introduce a fine-tuning approach incorporating row and column header information in the training instructions. We argue that this comparison with the zero-shot method is problematic. In zero-shot settings, due to the absence of a fine-tuning process, the format of the output tables is uncertain. For example, in ground-truth tables, cells that are left blank may be filled with terms such as "unknown" or "not mentioned" by the model, substantially impacting similarity-based metrics. Hence we modify the model's outputs under zero-shot settings before reporting the results. We evaluate the performance of ChatGPT with and without employing the T^3 method and also compare it to the fine-tuned LLaMA-7B model proposed by Tang et al. (2023). Results are detailed in Table 3. It is observable that the application of the T^3 method results in significant improvements across all metrics, with some measures outperforming the fine-tuned model. Further details on the experiments and analysis of the results are discussed in Appendix E.

6.3.2 Performance of T^3 on WIKI40B Dataset

We intend to evaluate the performance of our proposed approach in the text-to-table task without instructions and ground-truth tables. In line with the pioneering work of STRUCTSUM (Jain et al., 2024), we experiment on a randomly sampled set from the English section of WIKI40B dataset (Guo et al., 2020). As there is no ground-truth table for the text in the dataset, they leverage LLMs and propose AUTO-QA Coverage as an evaluation metric:

$$\text{Cov}(\mathcal{T}) = \frac{\sum_{i=1}^{|G(\mathcal{S})|} E_{(q_i,a_i)}[Q(\mathcal{T}, q_i)]}{|G(\mathcal{S})|}$$

where $G(\mathcal{S})$ is the list of Question-Answer pairs (q_i, a_i) generated by the LLM based on text $\mathcal{S}$, $Q(\mathcal{T}, q)$ is the LLM's answer to question q based on table $\mathcal{T}$, and $E_{(q,a)}[x]$ is the LLM's evaluation of whether answer a and x are equivalent for question q. On top of this evaluation, we add a step where an LLM is used to pre-screen each (q, a) pair based on text $\mathcal{S}$, filtering out any pairs where the question can not be correctly answered. This process further assures the quality of the QA pairs generated by G. We opt for ChatGPT as the LLM for evaluation and randomly sample 500 passages for the test dataset following Jain et al. (2024). We also introduce T^2(**T**ext-**T**uple) which treats the intermediary tuples from T^3 as a single table $\mathcal{T}$. We aim to investigate the extent of information loss during the conversion from tuples to tables through this configuration. Figure 5 shows the AUTO-QA Coverage of three methods. The curve indicates the percentage of generated tables meeting a given coverage threshold. Overall, T^3 demonstrates a

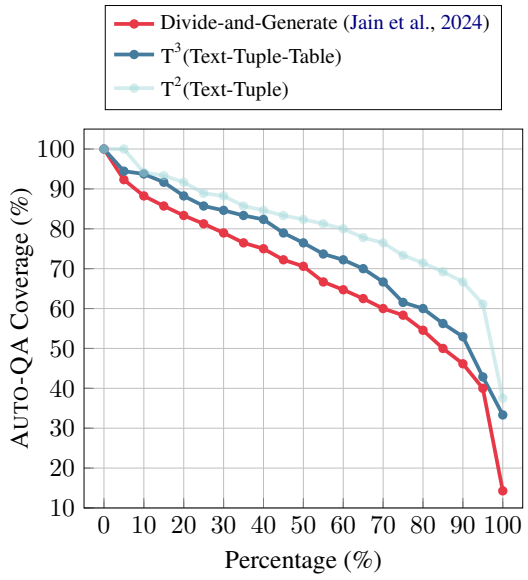

Figure 5: AUTO-QA coverage of the three methods. The point (P, C) means $P\%$ of the data can achieve a coverage of $C\%$ or higher measured using AUTO-QA.

substantial improvement over the prior Devide-and-Generate method (Jain et al., 2024). For example, when the coverage threshold is set to 70%, about 65% of data reach this threshold after applying T^3, compared to only 50% with the preceding approach. It is important to note that T^2 outperforms T^3, with 83% of data retaining the same coverage after tuple extraction. This suggests that information loss occurs during the transformation of raw tuples into structured tables. Exploring ways to mitigate this loss represents an essential area for further research. More details are discussed in Appendix F.

6.4 Case Studies

We present specific case studies on the outputs of different models on the LIVESUM dataset in Appendix G. These cases directly demonstrate the effectiveness of our proposed method. Additionally, we summarize some common errors of the model after applying T^3 and areas for improvement.

7 Related Work

Text-to-Table Generation Many studies have been proposed to perform text-to-table generation, converting it into sequence-to-sequence problems (Wu et al., 2022; Li et al., 2023c), or framing them as question-answering problems (Sundar et al., 2024). With the rise of LLMs, some research has also explored evaluating LLMs under fine-tuning or zero-shot settings, and it shows that fine-tuning yields highly effective results (Tang et al., 2023; Sundar et al., 2024). However, these methods employ datasets that only require the model to extract relevant information from text and populate tables, which significantly limits the scope of this task. Therefore, we introduce a new challenging dataset and propose a universal solution that greatly enhances the performance of LLMs under zero-shot setting.

LLMs for Information Extraction Information Extraction (IE) is critical and foundational for many downstream tasks in NLP. Many works have been conducted to leverage LLMs and provide effective solutions for IE tasks within a generative framework (Ma et al., 2023; Lu et al., 2023; Wan et al., 2023; Zhou et al., 2024). Recent progress in LLMs also has led to the development of unified frameworks that model various IE tasks and domains (Wang et al., 2023b; Sainz et al., 2024). This aligns with our intention to harness this capability to address general text-to-table generation tasks.

LLM Promptings Prompt engineering has been essential for enhancing LLMs and has demonstrated great success across a wide range of applications (Wei et al., 2022; Dua et al., 2022; Li et al., 2023a; Wang et al., 2024). Among the various prompting techniques, we find the decomposed prompting (Khot et al., 2022), which breaks down complex tasks into easier sub-tasks via prompting, highly effective for text-to-table generation. Jain et al. (2024) adopts this idea by breaking the text into small pieces for table generation. However, we argue that such a decomposition approach is impractical because text is not always easily divisible. For example, in our dataset, such division might result in adjacent sections describing the same event, causing errors. Hence we propose a more intuitive and broadly applicable task decomposition pipeline.

8 Conclusion

In this work, we introduce LIVESUM, a novel and challenging benchmark dataset for assessing the capability of models to integrate information in the text-to-table generation, along with a robust pipeline named T^3. Experimental results show that current LLMs underperform on our dataset in both fine-tuning and zero-shot settings; however, significant improvements are observed after applying our proposed T^3 pipeline. Our method can also be applied to any text-to-table dataset, enabling LLMs to outperform previous methods in zero-shot settings.

Limitations

Although our LIVESUM benchmark extensively evaluates the information integration capabilities of LLMs, we have not yet tested their performance in a few-shot setting. Despite the challenge posed by the token length of live commentary for few-shot settings, we reserve this aspect for future work. Furthermore, while our proposed T^3 pipeline significantly improves performance on several state-of-the-art LLMs, it cannot be effectively applied to LLMs that are deficient in tuple extraction capabilities, as it fails in the first stage and cannot proceed to the next phase. Developing methods that boost performance on such LLMs remains a valuable area for future research.

Ethics Statement

When constructing the LIVESUM dataset, we sample texts of live football match commentary from the open-access BBC Sports official website. We apply LLMs to paraphrase this live commentary and conduct manual reviews to ensure no harmful content is generated. We also anonymize the data using named entity recognition (NER) technology combined with player rosters. The datasets used in our experiments, STRUC-BENCH (Tang et al., 2023) and WIKI40B (Guo et al., 2020), are open-source, and all experiments adhere to their intended use for research purposes. Therefore, to the best of the authors' knowledge, we believe that this work introduces no additional risk.

References

Anthropic. 2023. Introducing claude 2.1. *Anthropic*.

Anthropic. 2024. Introducing the next generation of claude. *Anthropic*.

Junwei Bao, Duyu Tang, Nan Duan, Zhao Yan, Yuanhua Lv, Ming Zhou, and Tiejun Zhao. 2018. Table-to-text: Describing table region with natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Maximillian Chen, Alexandros Papangelis, Chenyang Tao, Seokhwan Kim, Andy Rosenbaum, Yang Liu, Zhou Yu, and Dilek Hakkani-Tur. 2023. PLACES: Prompting language models for social conversation synthesis. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 844–868, Dubrovnik, Croatia. Association for Computational Linguistics.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Yang Wang. 2020. Hybridqa: A dataset of multi-hop question answering over tabular and textual data. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1026–1036.

Dheeru Dua, Shivanshu Gupta, Sameer Singh, and Matt Gardner. 2022. Successive prompting for decomposing complex questions. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1251–1265, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. 2024. Deepseek-coder: When the large language model meets programming – the rise of code intelligence. *Preprint*, arXiv:2401.14196.

Mandy Guo, Zihang Dai, Denny Vrandecic, and Rami Al-Rfou. 2020. Wiki-40b: Multilingual language model dataset. In *LREC 2020*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Parag Jain, Andreea Marzoca, and Francesco Piccinno. 2024. Structsum generation for faster text comprehension. *Preprint*, arXiv:2401.06837.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. In *The Eleventh International Conference on Learning Representations*.

Hyunwoo Kim, Jack Hessel, Liwei Jiang, Peter West, Ximing Lu, Youngjae Yu, Pei Zhou, Ronan Bras, Malihe Alikhani, Gunhee Kim, Maarten Sap, and Yejin Choi. 2023a. SODA: Million-scale dialogue distillation with social commonsense contextualization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12930–12949, Singapore. Association for Computational Linguistics.

Hyunwoo Kim, Melanie Sclar, Xuhui Zhou, Ronan Bras, Gunhee Kim, Yejin Choi, and Maarten Sap. 2023b. FANToM: A benchmark for stress-testing machine

theory of mind in interactions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14397–14413, Singapore. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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

Jia Li, Ge Li, Yongmin Li, and Zhi Jin. 2023a. Structured chain-of-thought prompting for code generation. *Preprint*, arXiv:2305.06599.

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

Tong Li, Zhihao Wang, Liangying Shao, Xuling Zheng, Xiaoli Wang, and Jinsong Su. 2023c. A sequence-to-sequence&set model for text-to-table generation. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5358–5370, Toronto, Canada. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Di Lu, Shihao Ran, Joel Tetreault, and Alejandro Jaimes. 2023. Event extraction as question generation and answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1666–1688, Toronto, Canada. Association for Computational Linguistics.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023. Wizardcoder: Empowering code large language models with evol-instruct. *Preprint*, arXiv:2306.08568.

Yubo Ma, Yixin Cao, Yong Hong, and Aixin Sun. 2023. Large language model is not a good few-shot information extractor, but a good reranker for hard samples! In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10572–10601, Singapore. Association for Computational Linguistics.

MistralAI. 2024. Au large. *MistralAI*.

Varish Mulwad, Jenny Weisenberg Williams, Timothy W. Finin, Sharad Dixit, and Anupam Joshi. 2023. Towards semantic exploration of tables in scientific documents. In *ESWC Workshops*.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany. Association for Computational Linguistics.

OpenAI. 2022. Chatgpt: Optimizing language models for dialogue. *OpenAI*.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Oscar Sainz, Iker García-Ferrero, Rodrigo Agerri, Oier Lopez de Lacalle, German Rigau, and Eneko Agirre. 2024. Gollie: Annotation guidelines improve zero-shot information-extraction. *Preprint*, arXiv:2310.03668.

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

Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. 2024. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 645–654.

Anirudh Sundar, Christopher Richardson, and Larry Heck. 2024. gtbls: Generating tables from text by conditional question answering. *Preprint*, arXiv:2403.14457.

Xiangru Tang, Yiming Zong, Jason Phang, Yilun Zhao, Wangchunshu Zhou, Arman Cohan, and Mark Gerstein. 2023. Struc-bench: Are large language models really good at generating complex structured data? *Preprint*, arXiv:2309.08963.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti

Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Zhen Wan, Fei Cheng, Zhuoyuan Mao, Qianying Liu, Haiyue Song, Jiwei Li, and Sadao Kurohashi. 2023. GPT-RE: In-context learning for relation extraction using large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3534–3547, Singapore. Association for Computational Linguistics.

Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Doug Burdick, Darrin Eide, Kathryn Funk, Yannis Katsis, Rodney Michael Kinney, et al. 2020. Cord-19: The covid-19 open research dataset. In *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*.

Weiqi Wang, Tianqing Fang, Baixuan Xu, Chun Yi Louis Bo, Yangqiu Song, and Lei Chen. 2023a. CAT: A contextualized conceptualization and instantiation framework for commonsense reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13111–13140, Toronto, Canada. Association for Computational Linguistics.

Xiao Wang, Weikang Zhou, Can Zu, Han Xia, Tianze Chen, Yuansen Zhang, Rui Zheng, Junjie Ye, Qi Zhang, Tao Gui, Jihua Kang, Jingsheng Yang, Siyuan Li, and Chunsai Du. 2023b. Instructuie: Multi-task instruction tuning for unified information extraction. *Preprint*, arXiv:2304.08085.

Zilong Wang, Hao Zhang, Chun-Liang Li, Julian Martin Eisenschlos, Vincent Perot, Zifeng Wang, Lesly Miculicich, Yasuhisa Fujii, Jingbo Shang, Chen-Yu Lee, and Tomas Pfister. 2024. Chain-of-table: Evolving tables in the reasoning chain for table understanding. *Preprint*, arXiv:2401.04398.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, pages 38–45. Association for Computational Linguistics.

Jian Wu, Yicheng Xu, Yan Gao, Jian-Guang Lou, Börje Karlsson, and Manabu Okumura. 2023. TACR: A table alignment-based cell selection method for HybridQA. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6535–6549, Toronto, Canada. Association for Computational Linguistics.

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.

Derong Xu, Wei Chen, Wenjun Peng, Chao Zhang, Tong Xu, Xiangyu Zhao, Xian Wu, Yefeng Zheng, and Enhong Chen. 2023. Large language models for generative information extraction: A survey. *arXiv preprint arXiv:2312.17617*.

Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021. Bartscore: Evaluating generated text as text generation. *Advances in Neural Information Processing Systems*, 34:27263–27277.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *CoRR*, abs/2403.13372.

Wenxuan Zhou, Sheng Zhang, Yu Gu, Muhao Chen, and Hoifung Poon. 2024. Universalner: Targeted distillation from large language models for open named entity recognition. *Preprint*, arXiv:2308.03279.

Fengbin Zhu, Ziyang Liu, Fuli Feng, Chao Wang, Moxin Li, and Tat-Seng Chua. 2024. Tat-llm: A specialized language model for discrete reasoning over tabular and textual data. *Preprint*, arXiv:2401.13223.

Appendices

A Details of LIVESUM Dataset

This section extends Section 3, where we discuss additional details of dataset construction, and present detailed statistical information and a specific example of the dataset.

A.1 Live Commentary Generation

Paraphrasing We first utilize ChatGPT to paraphrase the original text to make it closer to the language used by the commentator while ensuring a certain degree of diversity. The prompting template used in this step is as follows:

Paraphrase Prompting Template

You are a football commentator, paraphrase the sentence and don't make it too long: `<Original Text>`

where `<Original Text>` represents the original live commentary on the webpage. After the transcription of each live commentary segment, we manually inspect whether the meaning of each sentence has been altered. If any issues are identified, we make manual revisions accordingly.

Data Anonymization We then perform the data anonymization. First, we obtain the list of players participating in each match from the BBC Sports official website. Then, using a combination of string matching and NER techniques (Qi et al., 2020), we match fully detected names to players on the list based on textual similarity. Each individual is assigned a unique number and recorded in the format `Player<number>(<team>)`, where `<number>` is a positive integer and `<team>` is the team that player belongs to. We anonymize the team names as "Home Team" and "Away Team". The inclusion of team names is intended to eliminate the need for the model to infer the team affiliation of each player, thus avoiding any potential interference with our evaluation of their information integration ability. We also manually inspect whether there is any missed anonymization. If any are found, we make manual revisions accordingly.

A.2 Summary Table Generation

Annotation We recruit five workers who are interested in football and are from English-speaking countries to annotate the events and employ a majority voting approach to resolve any disagreements. We randomly sample 100 cases with discrepancies and find that all errors are due to carelessness. Therefore, we conclude that the quality of the annotation quality is assured.

	Train Set	Test Set
# Instances	3,017	754
Average Text Length		
Words	1,258	1,250
Chars	6,852	6,816
Average Event Occurrence Frequency		
Goals	1.38	1.39
Shots	12.71	12.55
Fouls	10.60	10.57
Yellow Cards	1.74	1.76
Red Cards	0.04	0.03
Corner Kicks	5.25	5.24
Free Kicks	10.34	10.30
Offsides	1.86	1.84

Table 4: Average statistics in LIVESUM dataset.

Integration After manual annotation and review, the data are aggregated into several tuples, and we count the occurrence of each tuple to obtain the integrated result. Finally, we sequentially fill in the column headers of the table with the eight events, convert the team names to `Home Team` and `Away Team` for the row headers of the table, and then fill in the corresponding counts into the cells to generate the final summary table.

A.3 Statistics

Table 4 presents the statistical data regarding text length and the frequency of various events within the train and test sets of the LIVESUM dataset. It can be seen that the statistical metrics across the train set and test set are comparably uniform, consistent with our method of completely random division.

A.4 Example

Figure 6 presents an example of the live commentary and a summary table generated through the pipeline depicted in Figure 2.

B Implementation of T^3 on LIVESUM Dataset

This section is an extension of Section 4, where we provide description of the implementation details of the T^3 method applied to the LIVESUM dataset.

B.1 Instrction on LIVESUM Dataset

We present the instruction directives common to all experiments conducted on the LIVESUM dataset, which are some rules related to this task:

Instruction on LIVESUM Dataset

According to the live text, please count the number of: 1.goals, 2. shots, 3.fouls, 4.yellow cards, 5.red cards, 6.corner kicks, 7.free kicks, and 8.offsides for each team. Note that goals and saved attempts and blocked attempts and missed attempts are considered shots. Handball and dangerous play are also considered foul. The second yellow card is also considered a red card. Penalty is also considered as free kicks.

B.2 Text-to-Tuple Prompts

Based on the characteristics of the text in the Livesum dataset, we have defined the format of tuples as (*player name*, *team name*, *event*) or (*team name*, *event*). Subsequently, we define the following prompting template:

T^3 Text-to-Tuple Prompting Template

`<Instruction>`
Please extract all the relevant event from the following passage, output them in (*player name*, *team name*, *event*) or (*team name*, *event*) format. Constrain the event names to only the following options: 1.goals, 2.shots, 3.fouls, 4.yellow cards, 5.red cards, 6.corner kicks, 7.free kicks, and 8.offsides:
`<Text>`

where `<Instruction>` represents the text provided in Appendix B.1, and `<text>` is the live commentary text.

B.3 Information Integration Prompts

As discussed in Section 4.2, T^3 employs code generation as the default approach for information integration. The following is the prompting template we define, and Figure 7 presents an example of Python code generated by GPT-4.

T^3 Information Integration Prompting Template

Code Generation:
`<Instruction>`
Please develop a Python code to consolidate these tuples as specified:
`<Tuples>`

where `<Instruction>` represents the text provided in Appendix B.1, and `<Tuples>` is the tuples extracted in the previous step.

B.4 Tuple-to-Table Prompts

In this step, we present the format of the generated table and specify the following prompting template:

T^3 Tuple-to-Table Prompting Template

`<Instruction>`
Please only output a table with the team name in CSV format with 2 rows based on the following tuples:
`<Tuples>`

where `<Instruction>` represents the text provided in Appendix B.1, and `<Tuples>` is the tuples integrated by the code in the previous step.

B.5 Prompts of Variant Methods

In Section 6.2, two variants of the T^3 method are mentioned during the design of the ablation study. Here, we present the implementation details of these variants.

T^3-MERGED This method combines all three stages of T^3 into one, resulting in a single prompt:

T^3-MERGED Prompting Template

`<Instruction>`
Let's do the following things:
1. Extract all the relevant events from the following passage in (*player name*, *team name*, *event*) or (*team name*, *event*) format.
2. Integrate these tuples.
3. Output a table with 2 rows in CSV format.
`<Text>`

where `<Instruction>` represents the text provided in Appendix B.1, and `<text>` is the live commentary text.

T^3-DIRECT-EXECUTION This method replaces code generation with direct execution in the second stage of T^3. The prompts are as follows:

T^3-DIRECT Information Integration Prompting Template

`<Instruction>`
Please count all the information required and integrate these tuples:
`<Tuples>`

where `<Instruction>` represents the text provided in Appendix B.1, and `<Tuples>` is the tuples extracted in the previous step. The first and third steps remain unchanged and are described exactly as in Appendix B.2 and B.4.

C Details of Benchmark Configuration

This section serves as an extension of Section 5, providing additional details regarding the benchmark configuration.

C.1 Fine-Tune Setting

We use the open-source library named LLaMA-Factory[2] (Zheng et al., 2024) to fine-tune all models. LoRA (Hu et al., 2022) is used as the fine-tuning paradigm to accommodate our computational resources. The pre-trained weights are downloaded from the huggingface library (Wolf et al., 2020). We load the models with FP16 as the precision and optimize them with an Adam optimizer (Kingma and Ba, 2015). The learning rate is set to 5e-5 and the batch size is 32. The maximum length for the input and generated sentence concatenation is 3,500. We warm up the model with 3,000 steps and evaluate the model every 300 steps. A linear scheduler is also used. The LoRA rank is set to 8, and the α is set to 32.

C.2 Zero-Shot Setting

The zero-shot and chain-of-thought inference for LLaMA-2 models are conducted on eight V100 GPUs. When obtaining outputs from LLMs via APIs, we ensure deterministic results by setting the temperature to zero. In Table 5, we list the model names corresponding to different model types. Note that unless specifically stated otherwise, when a model type from the table is used, the model name corresponding to it in the table is the one being invoked.

Model Type	Model Name
Mistral Large	`mistral-large-2402`
ChatGPT	`gpt-3.5-turbo-0125`
GPT-4	`gpt-4-0613`
Claude 2.1	`claude-2.1`
Claude 3 Opus	`claude-3-opus-20240229`

Table 5: Model types and their corresponding names of LLMs used in our experiments.

C.3 Prompts

Below are the two prompting templates we used to evaluate the baseline models:

Baseline Prompting Template

Prompt w/o CoT
`<Instruction>`
Please output a table with 2 rows in CSV format according to the following live text:
`<Text>`

[2] https://github.com/hiyouga/LLaMA-Factory

Baseline Prompting Template

Prompt w/ CoT
`<Instruction>`
Let's think step by step! At last, please output a table with 2 rows in CSV format according to the following live text:
`<Text>`

where `<Instruction>` represents the text provided in Appendix B.1, and `<text>` is the live commentary text.

C.4 Result Parsing

As the table format in this experiment is fixed, most models can produce recognizable tables. We employ a comprehensive evaluation algorithm based on regular expressions to parse various types of output. For improperly formatted tables in the model's output, we filtered out those instances. This is because when the table format is disrupted, we are unable to obtain the meaning of each number in the output, and thus cannot calculate the metric. Moreover, this indicates that the model is even unable to perform the text-to-table generation task, which falls outside the scope of assessing its information integration abilities in the text-to-table process.

D Full Results of Ablation Study

In this section, we extend Section 6.2 and provide a comprehensive supplementary ablation study. We apply two variant methods, T^3-MERGED (T^3M) and T^3-DIRECT-EXECUTION (T^3D), proposed in Section 6.2, to all four LLMs capable of applying the T^3 method. The results of all experiments are listed in Table 6. In this table, we also provide a separate listing for each model, showcasing the average change in error rate for each method compared to the baseline. It is important to note that a lower error rate indicates better performance.

Impact of CoT Prompting Firstly, we observe that CoT prompting yields positive effects across all four LLMs, resulting in an overall reduction in error rate ranging from 0.3% to 2.4%. This further corroborates that CoT is a concise and effective prompting strategy.

Impact of T^3M Prompting After applying the T^3M method, it is observed that the performance of the Mistral Large and Claude 3 Opus deteriorates, particularly with a significant increase in error rate of 6.7% for Claude 3 Opus. This indicates that the models' ability to summarize the information internally is superior to the approach of extracting and

Model	Easy		Medium		Hard		Average		
	RMSE	ER	RMSE	ER	RMSE	ER	RMSE	ER	Δ_{ER}
Claude 2.1	1.014	10.08	2.581	63.99	4.621	90.58	3.171	57.16	
Claude 2.1 (CoT)	1.496	14.06	2.291	61.70	4.081	90.38	2.918	56.96	↓ 0.3%
Claude 2.1 (T^3M)	1.144	13.92	2.289	62.28	4.114	90.12	2.869	57.15	↓ 0.0%
Claude 2.1 (T^3D)	3.653	39.15	2.444	64.88	5.621	92.06	4.056	65.24	↑ 14.1%
Claude 2.1 (T^3)	**0.193**	**8.95**	**1.965**	**44.99**	**2.751**	**72.14**	**2.066**	**42.77**	↓ 25.2%
Mistral Large	**0.005**	**0.27**	2.385	52.45	2.712	84.62	2.209	47.44	
Mistral Large (CoT)	0.018	0.73	2.311	51.82	2.608	84.08	2.139	47.12	↓ 0.7%
Mistral Large (T^3M)	0.039	1.84	2.223	52.70	3.479	86.18	2.399	48.36	↑ 1.9%
Mistral Large (T^3D)	0.142	6.59	1.677	57.46	2.735	84.81	1.865	51.58	↑ 8.7%
Mistral Large (T^3)	0.191	8.82	**1.596**	**42.37**	**2.137**	**69.23**	**1.631**	**40.70**	↓ 14.2%
GPT-4	0.156	4.64	1.167	46.05	4.114	88.53	2.273	46.32	
GPT-4 (CoT)	0.154	4.38	1.163	45.86	3.981	88.73	2.225	46.20	↓ 0.3%
GPT-4 (T^3M)	**0.003**	**0.13**	1.419	43.22	2.458	82.76	1.621	42.34	↓ 8.6%
GPT-4 (T^3D)	0.087	4.47	1.124	40.13	232	81.45	1.418	41.55	↓ 10.3%
GPT-4 (T^3)	0.056	3.18	**0.854**	**25.83**	**1.219**	**46.22**	**0.929**	**25.27**	↓ 45.4%
Claude 3 Opus	0.078	2.52	1.617	51.36	3.713	88.06	2.253	48.33	
Claude 3 Opus (CoT)	**0.040**	**1.59**	1.642	49.60	3.265	87.87	2.079	47.17	↓ 2.4%
Claude 3 Opus (T^3M)	1.244	15.14	1.610	52.97	3.625	85.27	2.426	51.59	↑ 6.7%
Claude 3 Opus (T^3D)	0.327	9.66	1.315	46.43	1.924	79.50	1.432	45.50	↓ 5.9%
Claude 3 Opus (T^3)	0.081	5.30	**0.406**	**14.79**	**0.477**	**21.29**	**0.438**	**14.04**	↓ 70.9%

Table 6: The comprehensive ablation study results comparing the performance of different prompting methods. We **bold** the best results and underlined the second-best results.

merging information separately. On the contrary, GPT-4 demonstrates a significant reduction in error rate of 8.6% after applying T^3M, which indicates that the capabilities of GPT-4 are sufficiently robust to support the T^3M prompting approach and effectively execute all three steps. The change in error rate for Claude 2.1 is not significant; however, there is a notable 9.6% reduction in RMSE. This indicates that T^3M leads to a closer approximation of the true values in its results.

Impact of T^3D Prompting After applying T^3D, the error rates of Claude 2.1 and Mistral Large increase by 14.1% and 8.7%, respectively, indicating that they are not suitable for this prompting method. Comparing these results to those obtained with T^3, it can be inferred that the reason for the increase in error rate is likely due to their inferior ability to integrate information compared to the generated code, as the error rate significantly decreases when using code integration. On the other hand, GPT-4 and Claude 3 Opus achieved error rate reductions of 10.3% and 5.9%, respectively, under this prompting approach. This indicates that both models possess a certain level of ability to integrate information.

Impact of T^3 Prompting All four LLMs showed substantial improvements after applying the T^3 prompting method. Specifically, Mistral Large, Claude 2.1, and GPT-4 achieve average error rate reductions of 14.2%, 25.2%, and 45.4% respectively. Notably, Claude 3 Opus exhibits a remarkable error rate reduction of 70.9%. This strongly indicates the effectiveness of the proposed method. From the perspective of absolute error rates, under the T^3 method, the performance of the four models from best to worst is as follows: Claude 3 Opus, GPT-4, Mistral Large, and Claude 2.1. This ranking essentially reflects their capabilities in tuple extraction.

E Details of the Experiment on STRUCT-BENCH Table Dataset

This section is an extension of Section 6.3.1, in which we will introduce the evaluation criteria for this dataset, our implementation details, and an analysis of the results.

E.1 Evaluation Metrics

This benchmark employs nine evaluation metrics, five of which are traditional and applicable to text generation tasks: SacreBLEU (Post, 2018), ROUGE-L (Lin, 2004), BERTScore (Zhang et al., 2020), BLEURT (Sellam et al., 2020), and BARTScore (Yuan et al., 2021). These metrics can to some extent measure the similarity between the generated tables and the target tables. The last

Model	SacreBLEU	ROUGE-L	BERTScore	BLEURT	BARTScore	Content P-Score	Format P-Score	Content H-Score	Format H-Score
♦ *Zero-Shot*									
ChatGPT	77.58	86.11	96.75	64.66	-2.08	6.84	9.70	1.66	3.28
ChatGPT (T^3)	78.91	88.36	97.34	67.47	-1.90	7.29	**9.88**	1.68	3.63
GPT-4	87.26	**92.81**	98.15	77.08	-1.58	7.45	9.71	1.76	3.87
GPT-4 (T^3)	87.75	92.37	98.30	**79.80**	-1.60	7.60	9.74	**1.77**	**3.89**
♦ *Fine-Tuning*									
LLaMA-7B	**90.60**	88.98	**98.54**	66.07	**-0.69**	**7.69**	8.60	1.65	3.61

Table 7: Evaluation results on the test set of STRUC-BENCH Table dataset, consisting of nine metrics. We **bold** the best results and underlined the second-best results.

four metrics, P-score (Prompting Score) and H-score (Heuristical Score), are evaluation criteria proposed by Tang et al. (2023), which involve using ChatGPT to score the content and format of generated tables and applying manually devised rules to assess the content and format, respectively. For all evaluation metrics, higher numbers signify better performance.

E.2 Implementation Details

We conduct experiments on STRUCT-BENCH Table dataset (Tang et al., 2023) using ChatGPT and GPT-4 to assess the impact of T^3 on model performance. As described in Section 6.3.1, we first obtain the outputs of ChatGPT and GPT-4 using the provided prompting, removed terms such as "unknown" and "not mentioned" from the outputs, and then recalculate all metrics. This step involves using all instructions, prompting templates, and the code for calculating all metrics, all of which are sourced from the codebase[3] provided by Tang et al. (2023). We then apply our proposed T^3 method to both ChatGPT and GPT-4. We design the prompting template for the first step as follows:

T^3 on STRUCT-BENCH Table Dataset Text-to-Tuple Prompting Template

```
<Instruction>
You are now required to extract team and player information from the following input. Please focus on the table format and extract all relevant tuples in (team or player name, attribute, value) format:
<Text>
```

where `<Instruction>` is provided from the dataset, and `<text>` is the original text. Because the content of this dataset does not involve information integration, the second step does not alter the tuples from the first step. Therefore, we proceed directly to the third step of table generation, setting the following prompting template:

[3] https://github.com/gersteinlab/Struc-Bench

T^3 on STRUCT-BENCH Table Dataset Tuple-to-Table Prompting Template

```
<Instruction>
Based on the instructions and the following extracted tuples, please generate two tables according to the table format:
<Tuples>
```

where `<Instruction>` is provided in the dataset, and `<Tuples>` is the tuples extracted in the previous step.

E.3 Results Analysis

Table 7 presents all our experimental results, with the fine-tuning section quoting the state-of-the-art results (Tang et al., 2023). The results indicate that GPT-4 outperforms ChatGPT across all metrics. After applying the T^3 method, both ChatGPT and GPT-4 show improved performance, with ChatGPT experiencing a greater enhancement, suggesting the generalizability of the T^3 method. When compared with state-of-the-art fine-tuning methods, the zero-shot approaches perform better on some metrics. This indicates that the T^3 method still offers significant improvements for text-to-table tasks that do not require information integration, further demonstrating its broad applicability.

F Details of the Experiment on WIKI40B Dataset

This section is an extension of Section 6.3.2, in which we will discuss more details.

F.1 Evaluation Metrics

As introduced in Section 6.3.2, the evaluation criterion adopted for this dataset, AUTO-QA Coverage, is designed due to the absence of ground truth for the generated tables. Jain et al. (2024) propose using question answering as a medium and leverage LLMs to assess the quality of the generated tables. This metric not only measures how much information from the original text is covered by

the table but also checks the accuracy of the values within the table since incorrect content would also lead to errors in the QA. They also demonstrate through detailed experiments that AUTO-QA Coverage aligns with human evaluation standards, and thus we follow this metric in our experiments.

F.2 Implementation Details

Since Jain et al. (2024) samples 100 entries from the English version of WIKI40B (Guo et al., 2020) without disclosing which ones, we replicate their setting by sampling 500 texts to serve as the dataset for this experiment. For each passage, we utilize ChatGPT to generate 20 (question, answer) pairs that can be answered based on the text. Subsequently, we introduced a verification step where all (question, answer) pairs are re-fed into ChatGPT to confirm their correctness. Pairs that ChatGPT cannot accurately answer are filtered out. This approach enhances the accuracy of metric evaluation. Here, we present the prompts for T^3:

T^3 on WIKI40B Dataset
Text-to-Tuple Prompting Template

You are going to summarize a table for this passage, but the first step is to extract useful information. Output them in (*subject*, *attribute*, *value*) or (*subject*, *verb*, *object*) format: `<Text>`

where `<text>` is the original passage. It is noteworthy that, under this task setting, there are no instructions provided. Next, we feed the output tuples into the second step, where the LLM autonomously integrates the tuples:

T^3 on WIKI40B Dataset
Information Integration Prompting Template

Please integrate these tuples if necessary: `<Tuples>`

Finally, the tuples output from this step are fed into the prompt for the last step.

T^3 on WIKI40B Dataset
Tuple-to-Table Prompting Template

Summarize the triples below in one or multiple tables. Use the following format: Caption: A caption for the table you generate. It can be multiple lines. Table: A table in markdown format.
`<Tuples>`

The other prompts used in the experiments, such as those for generating question-answer pairs, assessing the correctness of answers, and the Divide-and-Generate method's prompting, all originate from Jain et al. (2024).

F.3 Results Analysis

The results of this experiment are presented in Figure 5. As discussed in Section 6.3.2, T^3 demonstrates higher AUTO-QA Coverage than the baseline, proving its ability to generate higher quality tables in text-to-table generation tasks without instructions. We also experiment with using the tuples extracted in the first step of T^3 directly as the generated tables and assessed their coverage, resulting in constructive findings. The coverage curve of T^2 is entirely above that of T^3, indicating a certain loss of information from tuple to table. Although this could also be due to insufficient table question answering capabilities leading to a decrease in metrics, it is necessary to employ relevant techniques (Wu et al., 2023; Wang et al., 2024) to mitigate the impact of this factor in future work. Nevertheless, the conclusion that T^3 improves the performance of the previous work remains sound.

G Case Studies

Figure 8 lists the outputs of four LLMs with and without the application of the T^3 method on the data shown in Figure 6. For the results not utilizing T^3, we can not easily analyze why it generates a range of large or small values. However, for the results using the T^3 method, we perform a detailed examination. We randomly sample 100 results generated by GPT-4 applying T^3 and conduct a spot check, finding that all errors originated from the first stage. Among these errors, 78% are due to missing event tuples, and 21% are due to wrong event tuples. Here, we present two representative examples.

Missing Event Tuple Example

Player18(Home Team) earns a free kick on the left wing after being fouled by Player20(Away Team).

```
(Player18, Home Team, Free Kick)
(Player20, Away Team, Foul)(missing)
```

Wrong Event Tuple Example

Player17(Home Team) from the Home Team draws a foul in the penalty area, resulting in a penalty conceded by Player33(Away Team).

```
(Player17, Home Team, Foul)(wrong)
(Player33, Away Team, Foul)
(Player17, Home Team, Free Kick)(missing)
```

Textual Live Commentary

Players are being announced for the lineup and getting ready for the game. The game is now underway with the start of the first half. Offside called on Home Team as Player5(Home Team) attempts a through ball to Player11(Home Team), who is caught offside. Player28(Away Team) makes a goal with a right-footed shot from the center of the box, giving the Away Team a 1-0 lead over the Home Team. Player8(Home Team) earns a free kick on the right side of the field. Player23(Away Team) commits a foul. Player9(Home Team) commits a foul. Player27(Away Team) earns a free kick in their own half. The Home Team earns a corner kick. Player5(Home Team)'s left footed shot from outside the box is saved in the bottom left corner after an assist from Player7(Home Team). Player26(Away Team) scores with a right-footed shot from outside the box, assisted by Player29(Away Team), Home Team 0, Away Team 2. The Home Team wins a corner kick. Player8(Home Team) commits a foul. Player21(Away Team) earns a free kick in their own half. Player29(Away Team) scores with a right-footed shot from the right side of the box, assisted by Player28(Away Team), Home Team 0, Away Team 3. Player7(Home Team) of the Home Team attempts a through ball, but Player11(Home Team) is flagged for being offside. Player29(Away Team) misses the goal with a high right footed shot from outside the box, assisted by Player26(Away Team). The Home Team wins a corner kick. Player10(Home Team) attempts a through ball, but Player11(Home Team) is offside for the Home Team. The Away Team wins a corner kick. Player5(Home Team) earns a free kick in their own half. Player21(Away Team) commits a foul. Player25(Away Team) fouls Player5(Home Team), who earns a free kick on the left wing. The Home Team wins a corner kick. Player10(Home Team) of the Home Team is caught offside after Player8(Home Team) attempts a through ball. At the end of the first half, the Home Team is trailing with a score of 0-3 against the Away Team. And we're back for the second half with the Home Team trailing 0-3 against the Away Team. Player2(Home Team) earns a free kick in their own half. Player23(Away Team) commits a foul. Player18(Home Team)'s header from the center of the box is saved in the bottom left corner. Player4(Home Team) commits a foul. Player29(Away Team) earns a free kick in their own half. Player4(Home Team) has received a yellow card for a reckless foul. Player29(Away Team) is currently delayed in the match due to an injury. The delay is finished and they are prepared to resume play. Player8(Home Team) is holding up the game due to an injury. The delay is finished and they are prepared to resume play. Player14(Home Team) commits a foul. Player23(Away Team) earns a free kick in the opponent's half. Player28(Away Team) missed the target with a shot from the right side of the box, with an assist from Player27(Away Team) after a quick counterattack. Player28(Away Team)'s right footed shot from the centre of the box was close, but missed to the right and then blocked. Player29(Away Team)'s shot from the center of the box is saved in the center of the goal after an assist from Player28(Away Team). The Away Team earns a corner kick. Player28(Away Team)'s shot from the center of the box is blocked with the assistance of Player23(Away Team). Player26(Away Team) attempts a through ball, but Player23(Away Team) is offside for the Away Team. Player14(Home Team) from the Home Team attempts a through ball, but Player11(Home Team) is flagged for being offside. The Home Team wins a corner kick. Player6(Home Team)'s close-range attempt is saved in the bottom left corner with an assist from Player3(Home Team)'s headed pass. Player18(Home Team) earns a free kick on the left wing after being fouled by Player20(Away Team). Player26(Away Team) attempts a through ball but Player20(Away Team) is offside for the Away Team. Player4(Home Team) commits a foul. Player28(Away Team) earns a free kick in their own half. Player27(Away Team)'s shot from the center of the box was blocked with the help of Player26(Away Team). Player6(Home Team) is being delayed in the match due to an injury. The delay is finished and they are prepared to resume play. Player18(Home Team)'s shot from outside the box with their right foot was too high, assisted by Player5(Home Team). Player6(Home Team) commits a foul on Player23(Away Team), resulting in a free kick being awarded on the left wing. The Away Team gets a corner kick. Player23(Away Team)'s left footed shot from outside the box following a corner is too high. Player25(Away Team) attempted a shot with his right foot from the right side of the box but it was too high, with an assist from Player28(Away Team). Player7(Home Team) earns a free kick in the opponent's half. Player36(Away Team) commits a foul. Player3(Home Team) misses the header to the left from the center of the box, assisted by Player16(Home Team) with a cross. Player28(Away Team) misses the target with a left-footed shot from long range. Player16(Home Team) attempts a through ball, but Player18(Home Team) is offside for the Home Team. The Home Team finishes the second half with a score of 0, while the Away Team has 3. Final score, Home Team 0, Away Team 3.

Summary Table

Team	Goals	Shots	Fouls	Yellow Cards	Red Cards	Corner Kicks	Free Kicks	Offsides
Away Team	3	12	6	0	0	3	6	2
Home Team	0	5	6	1	0	5	6	6

Figure 6: An instance within the LIVESUM dataset. The above section presents a complete textual live commentary that has undergone paraphrasing and data anonymization, while the table below represents a manually annotated summary table.

```python
# Initialize a dictionary to store the counts
team_stats = {}

# List of tuples
tuples = [('player1', 'home team', 'goal'),
          ('player2', 'home team', 'shot'),
          ('player3', 'away team', 'foul')]

# Loop through the tuples
for player, team, event in tuples:
    # If the team is not in the dictionary, add it
    if team not in team_stats:
        team_stats[team] = {'goals': 0, 'shots': 0, 'fouls': 0, 'yellow cards': 0, 'red cards': 0,
                            'corner kicks': 0, 'free kicks': 0, 'offsides': 0}

    # Update the counts based on the event
    if event == 'goal':
        team_stats[team]['goals'] += 1
        team_stats[team]['shots'] += 1
    elif event in ['shot', 'saved attempt', 'blocked attempt', 'missed attempt']:
        team_stats[team]['shots'] += 1
    elif event in ['foul', 'handball', 'dangerous play']:
        team_stats[team]['fouls'] += 1
    elif event == 'yellow card':
        team_stats[team]['yellow cards'] += 1
    elif event in ['red card', 'second yellow card']:
        team_stats[team]['red cards'] += 1
        if event == 'second yellow card':
            team_stats[team]['yellow cards'] += 1
    elif event == 'corner kick':
        team_stats[team]['corner kicks'] += 1
    elif event in ['free kick', 'penalty']:
        team_stats[team]['free kicks'] += 1
    elif event == 'offside':
        team_stats[team]['offsides'] += 1

# Print the counts for each team
for team, stats in team_stats.items():
    print(f"{team}: {stats}")
```

Figure 7: Python code generated by GPT-4 in response to the prompting as described in Appendix B.3.

Outputs from Claude 2.1

Results w/o T^3: RMSE=1.888, Error Rate=43.75%.

Team	Goals	Shots	Fouls	Yellow Cards	Red Cards	Corner Kicks	Free Kicks	Offsides
Away Team	3 ✓	13 ✗	10 ✗	0 ✓	0 ✓	3 ✓	5 ✗	4 ✗
Home Team	0 ✓	10 ✗	7 ✗	1 ✓	0 ✓	5 ✓	6 ✓	3 ✗

Results w/ T^3: RMSE=0.661, Error Rate=25.00%.

Team	Goals	Shots	Fouls	Yellow Cards	Red Cards	Corner Kicks	Free Kicks	Offsides
Away Team	3 ✓	12 ✓	4 ✗	0 ✓	0 ✓	3 ✓	6 ✓	2 ✓
Home Team	0 ✓	5 ✓	7 ✗	1 ✓	0 ✓	6 ✗	6 ✓	5 ✗

Outputs from Mistral Large

Results w/o T^3: RMSE=1.458, Error Rate=50.00%.

Team	Goals	Shots	Fouls	Yellow Cards	Red Cards	Corner Kicks	Free Kicks	Offsides
Away Team	3 ✓	10 ✗	7 ✗	0 ✓	0 ✓	3 ✓	10 ✗	2 ✓
Home Team	0 ✓	8 ✗	7 ✗	1 ✓	0 ✓	4 ✗	7 ✗	5 ✗

Results w/ T^3: RMSE=0.612, Error Rate=18.75%.

Team	Goals	Shots	Fouls	Yellow Cards	Red Cards	Corner Kicks	Free Kicks	Offsides
Away Team	3 ✓	12 ✓	6 ✓	0 ✓	0 ✓	3 ✓	4 ✗	2 ✓
Home Team	0 ✓	5 ✓	5 ✗	1 ✓	0 ✓	5 ✓	5 ✗	6 ✓

Outputs from GPT-4

Results w/o T^3: RMSE=1.785, Error Rate=31.25%.

Team	Goals	Shots	Fouls	Yellow Cards	Red Cards	Corner Kicks	Free Kicks	Offsides
Away Team	3 ✓	10 ✗	6 ✓	0 ✓	0 ✓	3 ✓	5 ✗	2 ✓
Home Team	0 ✓	11 ✗	9 ✗	1 ✓	0 ✓	5 ✓	7 ✗	6 ✓

Results w/ T^3: RMSE=0.433, Error Rate=18.75%.

Team	Goals	Shots	Fouls	Yellow Cards	Red Cards	Corner Kicks	Free Kicks	Offsides
Away Team	3 ✓	12 ✓	5 ✗	0 ✓	0 ✓	3 ✓	5 ✗	2 ✓
Home Team	0 ✓	5 ✓	6 ✓	1 ✓	0 ✓	5 ✓	5 ✗	6 ✓

Outputs from Claude 3 Opus

Results w/o T^3: RMSE=2.046, Error Rate=56.25%.

Team	Goals	Shots	Fouls	Yellow Cards	Red Cards	Corner Kicks	Free Kicks	Offsides
Away Team	3 ✓	18 ✗	7 ✗	0 ✓	0 ✓	2 ✗	7 ✗	3 ✗
Home Team	0 ✓	9 ✗	7 ✗	1 ✓	0 ✓	5 ✓	9 ✗	5 ✗

Results w/ T^3: RMSE=0.000, Error Rate=0.00%.

Team	Goals	Shots	Fouls	Yellow Cards	Red Cards	Corner Kicks	Free Kicks	Offsides
Away Team	3 ✓	12 ✓	6 ✓	0 ✓	0 ✓	3 ✓	6 ✓	2 ✓
Home Team	0 ✓	5 ✓	6 ✓	1 ✓	0 ✓	5 ✓	6 ✓	6 ✓

Figure 8: Case study analysis showing the outputs and evaluation metrics of Claude 2.1, Mistral Large, GPT-4, and Claude 3 Opus with and without the T^3 method on data shown in Figure 6.