Moneyball with LLMs: Analyzing Tabular Summarization in Sports Narratives

Ritam Upadhyay[1] * **Naman Ahuja**[1] * **Rishabh Baral** [1] **Aparna Garimella** [2] **Vivek Gupta** †[1]

[1]School of Computing and Augmented Intelligence, Arizona State University
[2]Adobe Research, India

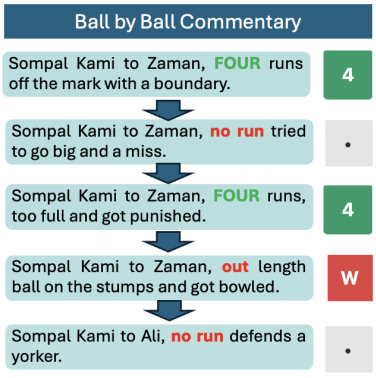
rupadh17@asu.edu, nahuja11@asu.edu, rbaral1@asu.edu, garimell@adobe.com,†vgupt140@asu.edu

Abstract

Large language model (LLM) approaches to tabular summarization rely on extensive prompt engineering, decomposition pipelines, or entity-level intermediate representations to achieve strong performance. While effective, these strategies are computationally expensive and offer limited insight into how well models maintain state over long, evolving narratives. We introduce SPORTABSET, a diagnostic benchmark for long-context tabular summarization across two complementary sports domains that require tracking multiple entities and aggregating statistics under domain-specific rules. Using SporTabSet, we systematically evaluate decomposition-based strategies across several long context LLMs. Results show that although decomposition substantially improves accuracy and numerical fidelity, gains stem mainly from dissecting multi-entity interference rather than improved local arithmetic. Robustness experiments further reveal high sensitivity to surface-level cues with structured failures, including hallucination, omission, and role confusion. Together, these findings identify consistent multi-entity memory as a key bottleneck in long-context table generation, motivating diagnostic evaluation as a prerequisite for scalable, efficient and reliable tabular summarization models.

1 Introduction

Tables are a central abstraction for summarizing and reasoning over complex information, enabling compact storage, comparison, and verification across domains such as scientific reporting, finance, and sports analytics. Early work on inferring structured summaries from dense, unstructured text, commonly referred to as Text-to-Table Generation or Tabular Summarization, primarily framed the task as an information extraction problem, emphasizing structural integrity, schema alignment,



	Bowler		Batsman	
Balls	Name	Runs	Name	Runs
1	Sompal Kami	4	Zaman	4
2	Sompal Kami	0	Zaman	0
3	Sompal Kami	4	Zaman	4
4	Sompal Kami	0	Zaman	OUT
5	Sompal Kami	0	Ali	0

Figure 1: Ball-by-ball commentary in cricket and updates to batsman and bowler tables after every ball.

and exact cell recovery from short or moderately sized inputs (Wu et al., 2022; Li et al., 2023; Sundar et al., 2024; Pietruszka et al., 2024).

More recent advances in large language models and structured generation interfaces, such as constrained decoding (Wang et al., 2025a), function calling, and tool-augmented agents (Schick et al., 2023), have made it increasingly feasible to produce well-formed structured outputs directly from LLMs (Hu and Wu, 2025; Li et al., 2025; Yao et al., 2022). This shift has enabled approaches that go beyond surface extraction, incorporating temporal reasoning over long contexts to generate summarized representations from evolving narratives (Hu et al., 2024). However, sustaining such behavior requires models to maintain evolving states of multiple attributes across many interacting entities, a challenge widely recognized in long-context and memory-augmented language modeling (Wang et al., 2023; Fang et al., 2024).

To mitigate this, recent methods externalize memory through iterative prompting (Ahuja et al.,

*These authors contributed equally
†Primary supervisor for this work

2025), structured memory representations (Deng et al., 2024a), or recursive summarization schemes (Wang et al., 2025b). While effective for tabular summarization, these designs often obscure how faithfully models maintain internal state over long contexts. Moreover, most table generation frameworks are evaluated only **one** dynamic benchmark (LiveSum)(Deng et al., 2024a), providing limited insight into how different modeling strategies generalize across narrative structures, table sizes, and entity regimes (Hu et al., 2024). As a result, it remains unclear *whether LLMs support long-context, multi-entity reasoning, which relies on surface regularities, and what trade-offs they incur in terms of accuracy, robustness, and computational cost*. This question is particularly compelling for LLMs, as they offer a unified interface for reasoning, memory, and generation, but lack explicit mechanisms for maintaining persistent, structured state.

To address these limitations, we introduce SPORTABSET, a diagnostic benchmark designed to systematically evaluate modeling strategies for dynamic text-to-table generation under long-context, multi-entity settings. SporTabSet spans two complementary sports domains: cricket and basketball, that expose distinct but related challenges in tabular summarization. Cricket requires maintaining and aggregating interdependent statistics across multiple evolving tables governed by explicit role-based game rules (batsmen and bowlers), while basketball emphasizes wide table construction with locally attributed updates across frequent entities. Together, these domains enable controlled analysis of how different modeling strategies handle entity tracking, temporal aggregation, and numerical consistency beyond short-context or purely extractive settings.

Our contributions are threefold:

- We introduce SPORTABSET, a multi-sport benchmark for long-context text-to-table generation, comprising over 6,500 cricket commentaries with programmatically derived ground truth and 1,988 unique basketball games aligned with official box scores (998 from 2021–2022 and 990 from 2025, expanded to 5,964 evaluation instances through label-invariant perturbations), along with a suite of label-invariant perturbations.
- We conduct a systematic comparison of modeling strategies for dynamic tabular summarization, from standard CoT (Wei et al., 2022) prompting to decomposition-based methods, entity-centric generation, and symbolic pipelines, analyzing their trade-offs across accuracy, numerical fidelity, and practical implications.
- We present a entity-centric robustness analysis that reveals structured failure modes such as hallucination, omission, role-confusion providing empirical evidence that maintaining multi-entity state is a primary bottleneck in long-context table generation with LLMs.

2 SporTabSet Benchmark

Cricket and basketball together form a complementary, entity-driven benchmark for evaluating large language models on dynamic text-to-table generation. In both sports, the task centers on resolving implicit references, performing numerical reasoning, and making consistent updates under a complex domain policy. This presents complementary challenges from previous SoTA benchmarks in this domain (Deng et al., 2024a) where each event induces only singular updates to a fixed table schema across the benchmark.

Despite these shared properties, the two sports impose distinct structural demands. Cricket presents long-horizon, multi-entity interactions with two distinct yet interdependent roles i.e. batsmen and bowlers, each requiring separate output tables and different logical reasoning: batsmen participate continuously across consecutive segments of play, while bowlers contribute episodically in distinct phases, necessitating phase-aware reasoning. Basketball commentary, in contrast, involves many agents operating within a single functional role, typically requiring the generation of one consolidated table covering a large roster of players (on average 24), where reasoning is dominated by dense, overlapping interactions occurring simultaneously rather than by role-specific or phase-structured temporal dependencies. With cricket stressing multi-table coordination and basketball requiring wide table construction, these datasets enable a systematic evaluation of LLM capabilities and robustness across diverse narrative and statistical regimes.

Dataset Collection and Preprocessing. We collect commentary–scorecard pairs from two sports domains: cricket from **Cricinfo**[1] and basketball from **ESPN**[2]. For cricket, we use ball-by-ball com-

[1] https://www.espncricinfo.com/
[2] https://www.espn.com/nba/scoreboard

mentary covering complete innings. To ensure chronological integrity and remove non-play content (e.g., weather updates, advertisements), we apply a monotonicity filter based on two criteria: **(i) Entity presence:** both the bowler and batsman names must co-occur in the line, separated by the standard delimiter pattern used across commentaries; and **(ii) Action presence:** the line must include a valid gameplay outcome such as *no run*, *FOUR*, *OUT*, or an extra (e.g., *1w*, *1nb*). Our base dataset comprises **498 ODI innings (2006–2025)**[3].

To analyze model performance across varying context lengths, we additionally create truncated samples by slicing the first x (120/180/240) balls of randomly selected samples, enabling systematic evaluation of temporal robustness and the effect of input length on model consistency. Each commentary is paired with two gold tables, a batsman scorecard and a bowler scorecard.

For basketball, we collect commentary for 1,988 unique games from ESPN, comprising 998 games from the 2021–2022 seasons and 990 games from the 2025 season. Each commentary is paired with the corresponding official final box score, which is scraped and used as the ground-truth table. For robustness experiments, these 1,988 games are expanded with label-invariant perturbations, yielding 5,964 basketball evaluation instances overall. Unlike cricket commentary, basketball play-by-play does not follow a sufficiently fixed textual pattern to support deterministic *rule-based* reconstruction using regular expressions. We therefore use the official box score as a verified reference table for per-player statistics such as points, rebounds, and assists. This should not be interpreted as the commentary lacking the relevant information: the events are described in the text, but recovering the final table requires tracking players and aggregating events across the full game rather than applying a simple parser. All data are sourced from publicly available pages under fair-use terms for academic research. A sample commentary and corresponding ground-truth tables are provided in Appendix F.1.

Ground Truth Creation. To generate the ground truth tables for truncated samples, that are not publicly available online, each innings is segmented into ball-level events using regular expressions based on Cricinfo's standard notation (e.g., *3.2: Starc to Kohli*). A domain-specific parser extracts outcomes such as runs, boundaries, wickets, and extras. Cricket extras (*wides, no-balls, byes, leg-byes*) are handled deterministically due to their asymmetric effects on batsman and bowler statistics, ensuring numeric consistency. Ball-level events are decomposed into constituent entities (Appendix F) and aggregated by player name to form the ground-truth batsman and bowler scorecards as shown in Table 18 and Table 17.

For basketball, we construct a single player-level scorecard per game. In contrast to cricket, where a stable notation enables regex-based reconstruction, basketball commentary is not regular enough for deterministic rule-based parsing into a full box score. We therefore scrape the official final box score corresponding to each commentary to serve as ground truth. The table contains 10 numeric columns of standard per-player statistics (e.g., points, rebounds, assists etc.) as shown in Table 19 and is aligned with commentary using game metadata. Importantly, the box score does not add hidden information beyond the commentary; rather, it serves as a verified reference for statistics that are recoverable from the full narrative but not automatically extractable with simple rules. This setup evaluates a model's ability to reconstruct a wide, entity-centric table from narrative text, even when the necessary evidence is distributed across many events.

Ground-Truth Validation. Since cricket ground-truth tables are generated programmatically, we conduct human verification to ensure benchmark correctness. For samples containing complete match commentary, we compare the derived scorecards against publicly available official summaries and observe a 99.7% exact match. For samples constructed via slicing, the authors manually validate 500 samples per setting, obtaining 99.3% exact match at the cell level and 100% accuracy for entity names (row headers).

For basketball, ground-truth tables are scraped directly from official sources. As the corresponding commentaries are not partially sliced, we perform manual verification on a random 10% subset of the 1,988 games to confirm 100% alignment between commentary and final box scores. This check is intended to verify that the official box score is consistent with the information present in the commentary and does not introduce hidden or extraneous statistics.

[3]One Day International of approximately 300 balls, played between national teams

3 Experiments

This section describes the evaluated LLMs, hybrid and decomposition-based baselines, and the metrics used to benchmark performance on SPORTABSET.

Models. We evaluate four state-of-the-art LLMs: **GPT-4.1** (OpenAI et al., 2024), **Gemini-2.5-Flash**, **Llama-3.3-70B-Instruct** (Grattafiori et al., 2024), and **Qwen-2.5-72B-Instruct** (Qwen et al., 2025). Since cricket commentaries average approximately 9k tokens, we restrict our evaluation to instruction-tuned models with context lengths exceeding 32k tokens that are adept at generating structured outputs.

Baselines. Prior work (Ball et al., 2024) demonstrates that Zero-Shot Chain-of-Thought (ZS-CoT) prompting consistently outperforms few-shot prompting for table generation tasks. Additionally, decomposition-based approaches that scale test-time compute have been shown to yield substantial performance improvements (Deng et al., 2024a; Ahuja et al., 2025). Motivated by these findings, we evaluate ZS-CoT alongside a set of decomposition-based methods that combine LLM-generated representations with rule-based parsers, and analyze their performance relative to the computational cost incurred. Specifically, we implement the following methods:

1. **ZS-CoT**: Models are provided with the cricket domain policy and instructed to reason step by step before producing the final scorecards.

2. **Divide & Generate (Jain et al., 2024)**: A two-stage pipeline in which the input commentary is partitioned into n chunks. Each chunk is processed independently using ZS-CoT to generate intermediate scorecards in parallel. These intermediate outputs are then deterministically merged to produce the final scorecards. We experiment with $n \in \{2, 4, 8\}$.

3. **Entity CoT**: This approach first prompts the LLM to extract the set of entities that defines the table schema. Each row is then generated using parallel LLM calls, with each call focusing on a single entity. While this increases input tokens (as each parallel call receives the full commentary), it simplifies the reasoning task by isolating entity-specific computation.

4. **Text–Tuple–Table**: Following (Deng et al., 2024a), this method first generates atomic tuples, one for each event of the form (Entity, Attribute, Value) signifying each update for the whole text. A rule-based compiler then aggregates these tuples to construct the final tables. This approach is the most computationally expensive, as generating atomic tuples over long-contexts significantly increases the number of output tokens per sample.

Evaluation Metrics. We evaluate models using a combination of continuous and discrete metrics. Root Mean Squared Error (RMSE) and Symmetric Mean Abs. Percentage Error (SMAPE) are used as quantitative metrics to measure numerical deviation between predicted and ground-truth table cells. SMAPE is particularly well-suited for commentary-driven scorecards, where many ground-truth cells are zero, as it avoids the disproportionate penalization of errors common with RMSE in sparse tables. In addition, we report Cell accuracy as a discrete correctness metric, measuring exact matches at the cell level to capture structural and symbolic fidelity of the generated tables.

A walk-through example for each method and the prompts used are provided in Appendix G.

3.1 Results and Analysis

In initial experiments, cricket commentaries exhibited unusually high performance, with **GPT-4.1 achieving 89% batsman cell accuracy**. Manual inspection revealed that this performance is largely driven by **explicit batsman dismissal summaries** (e.g., *run out (Chakabva) 5 (9b 0x4 0x6)*), which act as high-salience extractive cues. When all dismissal-related summaries are masked, accuracy drops sharply across models (e.g., GPT-4.1: 89% to 61%, Gemini: 86% to 49%; Table 9), indicating that batsman statistics are primarily inferred from narrative summaries rather than ball-by-ball reasoning. Notably, masking batsman cues also **degrades bowler accuracy**, suggesting that dismissal summaries function as shared structural anchors.

We therefore perform all critical baseline and perturbation analyses on the masked cricket datasets, which removes direct extractive cues; basketball contains no analogous summaries. We first quantify the overall impact of task decomposition relative to a zero-shot Chain-of-Thought (CoT) baseline, and then compare decomposition strategies in detail. All reported numbers are derived

Method	Cricket (Batsman)			Cricket (Bowler)			Basketball		
	Acc	RMSE	SMAPE	Acc	RMSE	SMAPE	Acc	RMSE	SMAPE
Llama-3.3-70B-Instruct									
CoT	39	18.18	29	42	10.27	29	29	3.83	62
Divide and Generate ($n = 2$)	40	15.19	21	40	12.34	31	33	6.15	57
Divide and Generate ($n = 4$)	44	14.81	17	42	10.98	25	47	4.66	36
Divide and Generate ($n = 8$)	48	12.93	12	45	5.83	**16**	58	2.84	24
EntityCoT	41	10.03	13	34	8.38	25	**62**	**1.55**	**19**
Text-Tuple-Table	**56**	**7.21**	**9**	**46**	**3.07**	**16**	57	1.69	27
GPT-4.1									
CoT	53	9.58	15	47	7.34	24	45	2.03	41
Divide and Generate ($n = 2$)	48	12.78	15	44	11.17	28	45	2.54	36
Divide and Generate ($n = 4$)	48	13.71	13	48	8.38	21	50	2.16	30
Divide and Generate ($n = 8$)	52	12.62	10	49	4.22	14	58	1.39	23
EntityCoT	55	7.14	10	57	3.99	14	67	**1.01**	**15**
Text-Tuple-Table	**64**	**4.23**	**5**	**59**	**2.15**	**12**	**68**	1.20	17
Gemini-2.5-Flash									
CoT	52	11.03	18	51	5.66	23	61	1.96	24
Divide and Generate ($n = 2$)	81	3.39	3	84	1.11	4	70	1.14	15
Divide and Generate ($n = 4$)	83	3.02	3	86	1.04	4	74	0.97	13
Divide and Generate ($n = 8$)	86	1.74	2	87	0.86	4	74	**0.96**	13
EntityCoT	89	**0.70**	**1**	82	1.17	**2**	72	1.03	13
Text-Tuple-Table	**94**	0.96	**1**	**89**	**0.59**	6	**75**	0.97	**12**

Table 1: Performance comparison on cricket and basketball datasets using Llama-3.3-70B-Instruct, GPT-4.1, and Gemini-2.5-Flash.

from the unified results in Table 1.

A. Overall Effect of Task Decomposition. We evaluate decomposition against a monolithic CoT baseline and find consistent improvements in accuracy and numerical fidelity across models and tasks.

Across models and tasks, **Divide-and-Generate** ($n = 8$) improves accuracy by **14.9 pp** and reduces RMSE by **37.8%**. **EntityCoT** achieves similar accuracy gains with larger RMSE reductions (**49.5%**), while **Text-Tuple-Table** yields the strongest improvements (**+20.6 pp** accuracy, **62.8%** RMSE reduction).

These gains are consistent across sports, with larger improvements in basketball, where reducing multi-entity interference is more critical.

At the same time, Table 1 should be interpreted as an *upper bound on achievable accuracy under regeneration*, not as a deployment-efficiency ranking. To make this distinction explicit, Appendix Table 16 co-reports first-pass failure rates, regeneration passes, and aggregate token cost over both successful and failed runs on a representative 100-sample-per-sport subset for Gemini-2.5-Flash and Llama-3.3-70B-Instruct.

B. Comparing Decomposition Strategies. We now compare three classes of decomposition strategies: Divide and Generate (input chunking), EntityCoT (player-wise chunking), and Text-Tuple-Table (event-level chunking). While **Divide-and-Generate** improves performance with the number of segments n, the gains are uneven across models and sports. At $n = 8$, Divide and Generate achieves a mean accuracy gain of **+14.9 pp** and a **37.8% RMSE reduction** across tasks. The gains are strongest for **Gemini-2.5-Flash**, where cricket accuracy improves by **+34 pp** and basketball accuracy by **+12 pp** over CoT. **Llama-3.3-70B-Instruct** shows moderate but consistent gains, including improvements of up to **+28 pp** on basketball. In contrast, **GPT-4.1** exhibits unstable behavior at smaller n.

EntityCoT performs *on par with or better than* Divide-and-Generate at $n = 8$, despite requiring fewer decomposition steps. It improves accuracy by **+15.1 pp** and reduces RMSE by **49.5%** relative to CoT. It is the best-performing method for basketball, achieving a mean gain of **+20.7 pp**. RMSE reductions are particularly pronounced: on basketball, EntityCoT reduces RMSE by **37–46%** for GPT-4.1 and Llama-3.3-70B-Instruct, while Gemini-2.5-Flash achieves over **69%** RMSE reduction. These results support our hypothesis that **multi-entity memory, rather than local arithmetic, is the primary bottleneck** in long-context table generation. Prior work on memory-augmented language models (Wang et al., 2023; Fang et al., 2024) has shown that retaining and retrieving long histories remains challenging even when local computation is straightforward. EntityCoT simplifies the reasoning problem and substantially reduces numerical drift, even when absolute accuracy gains are comparable to Divide-and-

Generate. Finally, **Text-Tuple-Table (T3)** delivers the strongest average performance across all methods. It achieves a mean accuracy gain of **+20.6 pp** and a **62.8% reduction in RMSE**, ranking first on both metrics overall. T3 is the top-performing method for all three models in aggregate, with particularly large gains for **Gemini-2.5-Flash** (**+31 pp** accuracy, **73%** RMSE reduction).

However, the appendix analysis shows that these gains do not come for free. In particular, T3 incurs substantially higher regeneration overhead once failed runs are counted, especially for Llama-3.3-70B-Instruct, whereas Divide-and-Generate and EntityCoT generally deliver more stable first-pass behavior for comparable improvements (Table 16).

C. Model-Specific Behavior. Models vary in their response to decomposition. **Gemini-2.5-Flash** shows the most consistent gains, while **Llama-3.3-70B-Instruct** improves steadily with decomposition. **GPT-4.1** exhibits less stable behavior, and T3 shows high variance and reliability issues. A more detailed analysis of cost and reliability is given in Appendix D.

Appendix Table 16 further clarifies this model-specific trade-off: Gemini-2.5-Flash combines strong gains with comparatively modest regeneration overhead, while Llama-3.3-70B-Instruct's strongest upper-bound results, particularly under T3, are paired with much higher first-pass failure rates and amortized token cost. This is precisely why we separate *best-valid performance* from *deployable efficiency* in our interpretation of Table 1.

While these methods represent an upper bound on achievable performance, their practical reliability can be potentially limited without additional safeguards, underscoring the need for foundational models for this task.

4 Entity-form Robustness

To assess whether models reason over input or rely on surface cues, we evaluate *form robustness* via three perturbations on the masked dataset for both cricket and basketball. The perturbations introduced are as follows:

A. Entity Substitution: Player names are replaced with *synthetic but plausible, human-sounding names* drawn from entirely different domains: for example, "Arun Hiachak to Lionel Cristiano" instead of "Muzarabani to Tamim."[4]

[4]Lionel Cristiano: **Lionel** Messi + **Cristiano** Ronaldo, two very famous soccer players.

	Batsman			Bowler		
	Anonymization (Original / Anonymized)					
	Acc	RMSE	SMAPE	Acc	RMSE	SMAPE
Gemini 2.5 Flash	52 / 53 +1	11.0 / 13.1 +2.09	18 / 20 +2	51 / 55 +4	5.6 / 4.7 -0.87	23 / 18 -5
Llama (70b)	39 / 34 -5	18.1 / 21.2 +3.04	29 / 36 +8	42 / 37 -4	10.2 / 11.4 +1.20	29 / 35 +6
Qwen (72b)	34 / 30 -4	21.6 / 34.8 +13.1	42 / 55 +13	40 / 37 -3	15.3 / 15.8 +0.50	36 / 40 +5
GPT 4.1	53 / 41 -12	9.7 / 13.4 +3.68	15 / 26 +11	47 / 43 -4	7.3 / 7.9 +0.66	24 / 28 +4
	OOD Entity Substitution (Original / Entity subs)					
	Acc	RMSE	SMAPE	Acc	RMSE	SMAPE
Gemini 2.5 Flash	52 / 58 +7	11.0 / 9.4 -1.61	18 / 13 -5	51 / 60 +9	5.6 / 3.9 -1.69	23 / 14 -9
Llama (70b)	39 / 38 -1	18.1 / 18.2 +2.60	29 / 28 -1	42 / 41 -0	10.2 / 9.3 -0.92	29 / 28 -1
Qwen (72b)	34 / 34 0	21.6 / 27.8 +6.21	42 / 42 0	40 / 38 -2	15.3 / 17.7 +2.47	36 / 38 +2
GPT 4.1	53 / 43 -10	9.7 / 9.4 -0.27	15 / 15 0	47 / 47 +1	7.3 / 5.7 -1.63	24 / 21 -3
	Entity Entanglement (Original / Entangled)					
	Acc	RMSE	SMAPE	Acc	RMSE	SMAPE
Gemini 2.5 Flash	52 / 51 -1	11.0 / 13.3 +2.34	18 / 21 +3	51 / 51 +0	5.6 / 5.4 -0.19	23 / 18 -5
Llama (70b)	39 / 34 -5	18.1 / 22.1 +3.99	29 / 40 +12	42 / 40 -1	10.2 / 10.8 +0.55	29 / 35 +6
Qwen (72b)	34 / 29 -4	21.6 / 31.3 +9.72	42 / 60 +18	40 / 34 -5	15.3 / 17.3 +2.07	36 / 48 +12
GPT 4.1	53 / 49 -4	9.7 / 11.5 +1.82	15 / 20 +5	47 / 44 -3	7.3 / 8.0 +0.72	24 / 29 +5

Table 2: Original vs. perturbed comparison of performance for cricket.

Models	Acc	RMSE	SMAPE
Anonymization (Original / Anonymized)			
Gemini	61 / 47 -14	1.96 / 3.15 +1.19	24 / 54 +30
Llama (70b)	29 / 23 -6	3.83 / 4.76 +0.93	62 / 82 +20
Qwen (72b)	30 / 27 -3	2.75 / 3.48 +0.73	67 / 79 +12
GPT-4.1	45 / 38 -7	2.03 / 2.56 +0.53	41 / 51 +10
OOD Entity Substitution (Original / Entity subs)			
Gemini	61 / 64 +3	1.96 / 1.75 -0.20	24 / 20 -4
Llama (70b)	29 / 27 -2	3.83 / 4.32 +0.48	62 / 67 +5
Qwen (72b)	30 / 23 -7	2.75 / 2.99 +0.23	67 / 75 +8
GPT-4.1	45 / 40 -5	2.03 / 2.14 +0.11	41 / 45 +4

Table 3: Original vs. perturbed comparison of performance for basketball.

B. Anonymization: Names of all player mentions are replaced with symbolic placeholder (e.g., `player1`, `player2`). This process mirrors the *delexicalization* strategy used in (Suntwal et al., 2019), to encourage the model to focus on relational and factual reasoning rather than lexical memorization.

C. Entity-Entanglement: This perturbation is specific to cricket, as basketball commentary does not involve distinct, role-dependent entities. Cricket commentary lines are paraphrased to break the standard "*<bowler> to <batsman>*" pattern by embedding player names within descriptive narrative (as shown in Appendix A). As a result, player roles are no longer explicitly stated and must be inferred from broader contextual cues. This requires models to reason holistically across multiple commentary lines to correctly assign roles and maintain

scorecard consistency.

4.1 Results and Analysis

Figure 2 and Tables 2, 3 show that perturbations do not simply degrade performance but systematically shift models between hallucination (overcounting) (Ji et al., 2023) and omission (undercounting), depending on the model, sport, role, and perturbation type. To capture directional error behavior, we define the Hallucination–Omission Index (HOI) as:

$$\text{HOI} = \text{Over-Count}\% - \text{Under-Count}\%$$

A. Entity Substitution (Out of Distribution). Entity substitution benefits Gemini in cricket but degrades performance in basketball. In cricket, replacing entities with OOD names improves Gemini's batsman accuracy by **6pp** and yields an approximate **15% reduction in RMSE**, while decreasing batsman HOI by about **5 percentage pp** (Figure 2, Table 2). In contrast, the same perturbation in basketball reduces accuracy by **21pp**, leads to an approximate **140% increase in RMSE**, and increases HOI by roughly **16 pp** (Figure 2; Table 3). This divergence suggests that cricket's summary-driven structure remains stable under name substitution, whereas basketball's dense, entity-linked updates are more sensitive to disrupted lexical anchors, consistent with prior observations on shortcut reliance (Kandpal et al., 2022).

B. Anonymization. Anonymization systematically increases hallucination, with model-specific effects. For Gemini batsmen in cricket, anonymization yields a marginal **+1pp** gain in accuracy while causing an approximate **19% increase in RMSE** and a **worsening in SMAPE**, indicating more correct cell placements but poorer numerical values (hallucinated precision; Table 2). In contrast, for GPT-4.1 batsmen, anonymization reduces accuracy by **12pp** and leads to an approximate **38% increase in RMSE**, while HOI remains negative with a small shift toward zero (about **+2pp**), reflecting omission-dominant failures due to withheld updates (Figure 2), a conservative strategy documented in prior work (Huang et al., 2023). In basketball, anonymization is catastrophic across models, with Gemini exhibiting a **27pp** accuracy drop alongside an approximate **180% increase in RMSE** and a large increase in hallucination (Table 3, Figure 2), confirming that removing lexical anchors in event-heavy commentary strongly amplifies hallucinated additions.

C. Entity Entanglement. Entity entanglement is the most disruptive perturbation in cricket and frequently flips error direction. For Qwen batsmen, paraphrasing increases hallucination substantially, raising HOI by approximately **+13pp** and causing an approximate **24% increase in overcount RMSE** (Table 2; Figure 2). For bowlers, the same perturbation can flip HOI from negative to positive, driven by event misattribution rather than token-level errors. Further Table 12 shows players being reported under incorrect player rows, demonstrating that breaking lexical role cues disrupts entity grounding and coreference, a known failure mode in contextualized models (Joshi et al., 2019).

Joint analysis of HOI with RMSE and SMAPE reveals three recurring model behaviors. **Hallucination-prone** models (Gemini, often Qwen) show positive HOI and rising numeric error under identity or role disruption, with cases of hallucinated precision (e.g., Gemini batsmen exhibit a **5pp** HOI decrease under anonymization while RMSE increases). **Omission-dominant** models (GPT-4.1) maintain negative HOI across perturbations and trade accuracy for conservative numeric estimates. **Unstable** models (Qwen, Llama) exhibit large HOI swings under paraphrase or anonymization, indicating brittle entity–role grounding.

Accuracy alone obscures these differences: GPT-4.1 batsmen can lose **12pp** accuracy while numeric error improves, whereas Gemini batsmen under anonymization gain **1pp** accuracy with worse RMSE/SMAPE. These patterns are consistent across cricket and basketball (Tables 2–3; Figure 2).

Temporal Robustness. We also studied **how performance varies as the input horizon grows** and whether this temporal effect depends on the presence of summary cues using partial inputs, results of which has been highlighted in Appendix C. However, basketball games are kept intact without any partial inputs.

5 Effect of Memorization

Since ESPN is a widely scraped web source, we probe potential data contamination using chronological splits. Beyond detecting contamination, this setup allows us to study how pretraining exposure influences performance in long-context, multi-entity reasoning tasks. We perform this analysis

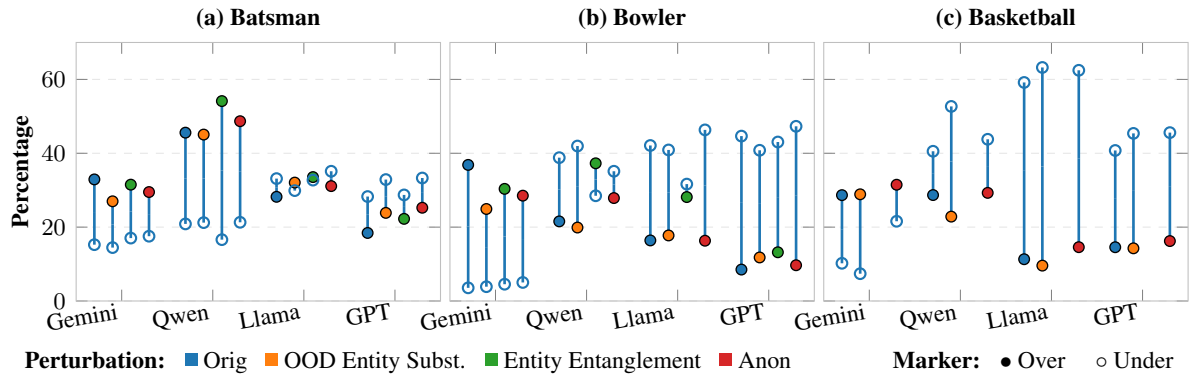


Figure 2: Over- and under-count percentages in cells for batsman (a), bowler (b), and basketball (c).

across both cricket and basketball to assess whether these effects are consistent across domains. For cricket, where commentaries span a long temporal range (2006–2025), we partition samples into pre- and post-2025 sets (January–May 2025), treating the latter as a held-out proxy for post-training evaluation for most models. All models are evaluated with identical prompting across splits. Reported knowledge cutoffs are GPT-4.1 (July 2024), Gemini-2.5-Flash (January 2025), Qwen-2.5-72B-Instruct (September 2024), and Llama-3.3-70B-Instruct (December 2023).

This setup reveals patterns consistent with exposure-driven effects in cricket. GPT-4.1 shows an approximately **12pp** drop in batsman accuracy on post-2025 data, accompanied by only a small RMSE change, indicating increased withholding of uncertain updates rather than consistent state maintenance. In contrast, Gemini-2.5-Flash improves by approximately **3pp** in batsman accuracy, with substantial reductions in numerical error (batsman RMSE **6.06** to **4.90**, bowler RMSE **3.90** to **1.63**). Structural consistency also improves: missing and extra rows drop (9% to 2%) on newer data, suggesting improved entity-level state tracking rather than memorized recall. These trends indicate that exposure effects vary across models. These trends persist under EntityCoT prompting: GPT-4.1 exhibits a large RMSE reduction (9.58 to 6.34) but only modest accuracy gains. Across other baselines, GPT-4.1 consistently shows the smallest accuracy improvement from CoT relative to other models (Table 1). Notably, these trends persist across prompting strategies, including EntityCoT, indicating that while task decomposition mitigates multi-entity interference, it does not eliminate biases introduced by pretraining exposure.

To test whether this effect persists even when commentary-based cues are removed, we include

Model	Avg. RMSE	Avg. Accuracy
Cricket control: names + venue/date		
	RMSE (Old / New)	**Acc (Old / New)**
Gemini	21.5 / 23.3	42 / 17
2.5 Flash	+1.8	-24
GPT	12.7 / 22.7	58 / 17
4.1	+10.0	-41
Llama	27.3 / 25.9	26 / 22
(70b)	-1.4	-4
Qwen	20.7 / 25.7	18 / 16
(72b)	+5.0	-3
Basketball control: title + boxscore schema		
	RMSE (Old / New)	**Acc (Old / New)**
Gemini	2.9 / 3.8	37 / 42
2.5 Flash	+0.9	+5
GPT	2.4 / 2.8	38 / 34
4.1	+0.4	-3
Llama	3.0 / 3.6	32 / 28
(70b)	+0.6	-4
Qwen	3.0 / 3.4	28 / 23
(72b)	+0.4	-5

Table 4: Controlled memorization probes without commentary. In cricket, models receive only batsman names, tournament, venue, and date and must reconstruct the final batsman scorecard. In basketball, models receive the game title and a blank boxscore schema with row and column headers already present and must fill in the stat cells.

controlled memorization probes for both sports. To isolate memorization, we introduce controlled probes without commentary. In cricket, models receive only batsman names, tournament, venue, and date, and must reconstruct the final batsman scorecard. In basketball, models receive the game title together with the boxscore schema, row and column headers, and the player/team rows already listed, while all stat cells are left blank for completion. Because no narrative evidence is provided, performance differences in this setting cannot be attributed to reasoning over input, isolating the effect of prior exposure.

The controlled cricket results in Table 4 show a substantial old-versus-new gap even without commentary. Older cricket matches remain markedly easier to reconstruct for several models: GPT-4.1 shows an approximately **41pp** accuracy advantage

on older matches together with a **9.98** lower RMSE, while Gemini-2.5-Flash shows an approximately **24pp** accuracy advantage and a **1.74** lower RMSE. Qwen also retains a smaller but consistent advantage on older matches. Llama-3.3-70B-Instruct is the main exception, showing only a modest accuracy advantage on older data while achieving slightly lower RMSE on newer matches.

The basketball control shows weaker and more model-specific chronological effects. Gemini-2.5-Flash improves in accuracy (**37%** to **42%**) but with worse RMSE (**2.88** to **3.77**), while GPT-4.1, Llama-3.3-70B-Instruct, and Qwen-2.5-72B-Instruct all degrade on newer games, with accuracy drops of **3–5** points and RMSE increases of **0.36–0.57**. Overall, exposure effects are strong in cricket but weaker in basketball. These findings complement our perturbation results, suggesting that both memorization and reliance on surface-form cues may contribute to observed behavior. Together, these findings show that memorization is not a uniform confound but a model- and domain-dependent factor that can either help or hinder performance. For example, GPT-4.1 exhibits a substantial accuracy drop on newer cricket data (approximately **12pp**), while Gemini-2.5-Flash shows improvements under the same setting. In basketball, these effects are weaker and more model-specific. While these probes provide practical diagnostics, the observed variability across models and sports suggests that exposure effects interact with multi-entity reasoning rather than simply inflating performance.

6 Related Work

A large body of work demonstrates that high average performance can often mask underlying fragilities and reliance on spurious correlations. Methodologies have evolved to include behavioral testing with checklists to evaluate specific linguistic capabilities (Ribeiro et al., 2020), adversarial and counterfactual examples to assess sensitivity to minor input perturbations (Jia and Liang, 2017; McCoy et al., 2019; Kaushik et al., 2020), and analyses of long-context reasoning to identify performance degradation over extended inputs (Liu et al., 2024). These diagnostic approaches are essential for understanding model limitations and are increasingly important for information-centric tasks, where reliability and faithfulness to source material are paramount.

The task of text-to-table (T2T) generation has similarly matured from a structured information extraction (IE) problem to a more complex reasoning challenge. Recent approaches have tackled increasingly complex settings where table values must be derived through aggregation, counting, and tracking state changes over long, narrative-style texts. Systems now address nuanced domains like sports analytics (Hu et al., 2024) and finance (Zhu et al., 2024), generate complex hierarchical tables (Cheng et al., 2021) and employ sophisticated schema-guided pipelines or iterative event extraction (Ahuja et al., 2025; Deng et al., 2024b). As T2T systems take on these more demanding reasoning tasks, the need for robust evaluation, drawing on the principles of model auditing, becomes indispensable for ensuring their reliability.

7 Conclusion

We present SPORTABSET, a benchmark for long-context text-to-table generation, and evaluate how LLMs summarize evolving sports narratives into structured representations.

Our findings lead to three key takeaways. First, task decomposition consistently improves performance, and its benefits arise primarily from mitigating multi-entity interference while enhancing local arithmetic or counting ability across smaller inputs also helpful. Second, models remain brittle under semantics-preserving perturbations, revealing that surface-level cues and memorized patterns play a substantial role in apparent reasoning success. Third, methods that achieve the strongest aggregate performance often incur high costs in reliability and implementation complexity, highlighting a gap between benchmark accuracy and deployable systems.

Taken together, these results suggest that progress in long-context tabular summarization will require models that can natively maintain structured, entity-consistent state over extended inputs, rather than relying on external memory traces, or brittle surface cues. SporTabSet provides a controlled setting for studying these challenges, and we hope it will inform the development of foundational models and evaluation paradigms that treat multi-entity memory and robustness as first-class objectives in structured generation.

Limitations

Our study is intentionally diagnostic and scoped to understanding long-context text-to-table generation under controlled, multi-entity settings. While

SporTabSet spans two complementary sports domains, both cricket and basketball follow structured narrative conventions and benefit from dense, verifiable supervision. Extending this analysis to other high-impact domains such as finance, healthcare, or scientific reporting remains challenging due to limited public availability of long-form narratives paired with reliable, fine-grained ground truth. Future work that incorporates privacy-preserving data releases or synthetic-but-faithful benchmarks could help bridge this gap.

Our evaluation focuses on stateful numerical tables with fixed schemas and does not consider more complex table structures, such as hierarchical headers, merged cells, temporal indices, or heterogeneous data types (e.g., categorical fields or free-text cells). These structural dimensions introduce additional challenges that are orthogonal to multi-entity state tracking and remain unexplored in this work. Incorporating such table representations would enable a more comprehensive assessment of structured generation capabilities.

Although we explicitly probe memorization effects through chronological splits and entity-level perturbations, we do not attempt to mitigate biases introduced by pretraining on widely scraped sources. Moreover, samples in the old and new splits may differ in inherent difficulty due to stylistic or contextual variation. We partially control for this by maintaining comparable input lengths, schema structure, and event density across splits, but residual confounds may remain. Addressing memorization at the model or data level rather than diagnosing its effects remains an important direction for future research.

Finally, our analysis emphasizes correctness, numerical fidelity, and robustness, but does not directly evaluate downstream utility or human trust in generated tables. Nor do we propose new architectural mechanisms for persistent memory; instead, we focus on characterizing the limitations of existing prompting and decomposition strategies. Future work could translate these findings into architectural or training interventions that enable models to natively maintain structured, entity-consistent state over long contexts, reducing reliance on external prompting scaffolds or symbolic post-processing.

Ethics Statement

All data used in this study was collected from publicly accessible sports commentary on ESPNcricinfo[5]. The dataset consists of ball-by-ball descriptions and player names that are already publicly available in broadcast and written formats, and does not include private or sensitive personal information.

The data is used solely for academic research to evaluate language models on text-to-table generation. To mitigate concerns around memorization and identity reliance, we include robustness experiments such as anonymization and entity substitution. We do not attempt to generate content that could harm individuals or organizations, and all outputs are intended for controlled evaluation under fair-use research guidelines.

Licence Statement

The benchmark artifacts (ground-truth tables, perturbations, and evaluation code) are released under the Creative Commons Attribution–NonCommercial–ShareAlike 4.0 International (CC BY-NC-SA 4.0) license. This permits use, distribution, and adaptation for academic research purposes only, provided that appropriate credit is given and derivative works are shared under the same terms.

The dataset includes structured representations derived from commentary text publicly available on ESPNcricinfo. Original commentary remains under ESPNcricinfo copyright and is redistributed here solely under fair-use provisions for non-commercial research. Commercial use of these artifacts is strictly prohibited.

References

Naman Ahuja, Fenil Bardoliya, Chitta Baral, and Vivek Gupta. 2025. Map&make: Schema guided text to table generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 30249–30262, Vienna, Austria. Association for Computational Linguistics.

Thomas Ball, Shuo Chen, and Cormac Herley. 2024. Can we count on llms? the fixed-effect fallacy and claims of gpt-4 capabilities. *arXiv preprint arXiv:2409.07638*.

Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and

[5]https://www.espncricinfo.com/

Dongmei Zhang. 2021. Hitab: A hierarchical table dataset for question answering and natural language generation. *arXiv preprint arXiv:2108.06712*.

Zheye Deng, Chunkit Chan, Weiqi Wang, Yuxi Sun, Wei Fan, Tianshi Zheng, Yauwai Yim, and Yangqiu Song. 2024a. Text-tuple-table: Towards information integration in text-to-table generation via global tuple extraction. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9300–9322, Miami, Florida, USA. Association for Computational Linguistics.

Zheye Deng, Chunkit Chan, Weiqi Wang, Yuxi Sun, Wei Fan, Tianshi Zheng, Yauwai Yim, and Yangqiu Song. 2024b. Text-Tuple-Table: Towards Information Integration in Text-to-Table Generation via Global Tuple Extraction. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9300–9322, Miami, Florida, USA. Association for Computational Linguistics.

Junjie Fang, Likai Tang, Hongzhe Bi, Yujia Qin, Si Sun, Zhenyu Li, Haolun Li, Yongjian Li, Xin Cong, Yankai Lin, Yukun Yan, Xiaodong Shi, Sen Song, Zhiyuan Liu, and Maosong Sun. 2024. Unimem: Towards a unified view of long-context large language models. In *First Conference on Language Modeling*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Ruike Hu and Shulei Wu. 2025. Rl-struct: A lightweight reinforcement learning framework for reliable structured output in llms. *Preprint*, arXiv:2512.00319.

Yebowen Hu, Kaiqiang Song, Sangwoo Cho, Xiaoyang Wang, Hassan Foroosh, Dong Yu, and Fei Liu. 2024. Sportsmetrics: Blending text and numerical data to understand information fusion in llms. *Preprint*, arXiv:2402.10979.

Wenlong Huang, Fei Xia, Dhruv Shah, Danny Driess, Andy Zeng, Yao Lu, Pete Florence, Igor Mordatch, Sergey Levine, Karol Hausman, and brian ichter. 2023. Grounded decoding: Guiding text generation with grounded models for embodied agents. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Parag Jain, Andreea Marzoca, and Francesco Piccinno. 2024. STRUCTSUM generation for faster text comprehension. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7876–7896, Bangkok, Thailand. Association for Computational Linguistics.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12).

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2275–2285, Copenhagen, Denmark. Association for Computational Linguistics.

Mandar Joshi, Omer Levy, Daniel S. Weld, and Luke Zettlemoyer. 2019. Bert for coreference resolution: Baselines and analysis. *ArXiv*, abs/1908.09091.

Nikhil Kandpal, Eric Wallace, and Colin Raffel. 2022. Deduplicating training data mitigates privacy risks in language models. *ArXiv*, abs/2202.06539.

Divyansh Kaushik, Eduard Hovy, and Zachary Lipton. 2020. Learning the difference that makes a difference with counterfactually-augmented data. In *International Conference on Learning Representations*.

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. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5358–5370.

Yiqi Li, Yusheng Liao, Zhe Chen, Yanfeng Wang, and Yu Wang. 2025. DICE: Structured reasoning in LLMs through SLM-guided chain-of-thought correction. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 6951–6966, Suzhou, China. Association for Computational Linguistics.

Nelson F. Liu, Kevin Levonian, Iz Beltagy, Kyle Lo, and Noah A. Smith. 2024. Lost in the middle: How language models use long contexts. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7626–7644, Mexico City, Mexico. Association for Computational Linguistics.

R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2240–2250, Minneapolis, Minnesota. Association for Computational Linguistics.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Michał Pietruszka, Michał Turski, Łukasz Borchmann, Tomasz Dwojak, Gabriela Nowakowska, Karolina Szyndler, Dawid Jurkiewicz, and Łukasz Garncarek. 2024. Stable: Table generation framework for encoder-decoder models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2454–2472.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. CheckList: Beyond accuracy in NLU evaluation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4902–4912, Online. Association for Computational Linguistics.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, and 1 others. 2023. Toolformer: Language models can teach themselves to use tools. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

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

Sandeep Suntwal, Mithun Paul, Rebecca Sharp, and Mihai Surdeanu. 2019. On the importance of delexicalization for fact verification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3413–3418, Hong Kong, China. Association for Computational Linguistics.

Darren Yow-Bang Wang, Zhengyuan Shen, Soumya Smruti Mishra, Zhichao Xu, Yifei Teng, and Haibo Ding. 2025a. Slot: Structuring the output of large language models. *Preprint*, arXiv:2505.04016.

Qingyue Wang, Yanhe Fu, Yanan Cao, Shuai Wang, Zhiliang Tian, and Liang Ding. 2025b. Recursively summarizing enables long-term dialogue memory in large language models. *Preprint*, arXiv:2308.15022.

Weizhi Wang, Li Dong, Hao Cheng, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. 2023. Augmenting language models with long-term memory. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*.

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.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*.

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 financial tabular and textual data. In *Proceedings of the 5th ACM International Conference on AI in Finance*, pages 310–318.

A Data Perturbations

Original	OOD Entity Sub.	Anon.	Entangle -ment
Muzarabani to Tamim, no run fullish ball on middle and leg, clipped to square leg.	Vinicius Alcaraz to Matt Inge-brigtsen, no run fullish ball on middle and leg, clipped to square leg.	Player3 to Player7, no run fullish ball on middle and leg, clipped to square leg.	Spotlight swings toward Muzara-bani / Tamim; no run fullish ball on middle and leg, clipped to square leg.
Muzarabani to Tamim, no run full and straighter on off this time. Driven down the ground with the full face of the bat.	Vinicius Alcaraz to Matt Inge-brigtsen, no run full and straighter on off this time. Driven down the ground with the full face of the bat.	Player3 to Player7, no run full and straighter on off this time. Driven down the ground with the full face of the bat.	Action with Tamim and Muzara-bani: no run full and straighter on off this time. Driven down the ground with the full face of the bat.
Muzarabani to Tamim, no run past the outside edge to the batter. Fullish length in the off-stump channel, as Tamim presents his bat playing in the line and the ball shoots off the pitch.	Vinicius Alcaraz to Matt Inge-brigtsen, no run past the outside edge to the batter. Fullish length in the off-stump channel, as Matt Inge-brigtsen presents his bat playing in the line and the ball shoots off the pitch.	Player3 to Player7, no run past the outside edge to the batter. Fullish length in the off-stump channel, as Player7 presents his bat playing in the line and the ball shoots off the pitch.	Between Tamim and Muzara-bani: no run past the outside edge to the batter. Fullish length in the off-stump channel, as Tamim presents his bat playing in the line and the ball shoots off the pitch.

Table 5: Comparison of cricket commentary variations across different transformation types. Highlighted text indicates specific differences from the original commentary in terms of entity substitution (player names), structural changes, and semantic modifications.

We induce three perturbations to study the robustness of the models. **Anonymization** replaces the named entities for bowlers and batsmen with generic placeholders like "*Player1*" and "*Player2*". **Out-of-distribution entity substitution** replaces the entities with fabricated names generated using amalgamations of names from sports other than cricket. **Entity entanglement** deliberately obscures entity roles to challenge role identification. In the original cricket commentary, predictable structural patterns clearly distinguish batsmen from bowlers (e.g., "Sharma bowls to Kohli" vs. "Kohli faces Sharma"). This perturbation disrupts these cues by using ambiguous phrasing and reversed sentence structures, requiring models to maintain holistic contextual understanding rather than relying on positional heuristics. Examples contrasting the original, anonymized, OOD entity-substituted, and entity-entangled versions are shown in Table 5 for cricket, while the equivalent perturbations are shown for basketball in Table 6. Note that this perturbation was applied only to cricket commentary, as basketball play-by-play lacks the same structured role distinctions between offensive and defensive players.

Original	OOD Entity Sub.	Anon.
Nic Claxton vs. Brook Lopez (Giannis Antetok-ounmpo gains possession)	Madison Jackson vs. Dustin Bolt (Rafael Jefferson gains possession)	Player35 vs. Player4 (Player9 gains possession)
Giannis Antetok-ounmpo misses 11-foot hook shot	Rafael Jefferson misses 11-foot hook shot	Player9 misses 11-foot hook shot
James Harden blocks Giannis Antetok-ounmpo's two point shot	Auston Sanders blocks Rafael Jefferson's two point shot	Player 14 blocks Player 9's two point shot

Table 6: Comparison of basketball commentary variations across different transformation types. Highlighted text indicates specific differences from the original commentary in terms of entity substitution (player names), structural changes, and semantic modifications.

A.1 Effect on Memorization

	Batsman			Bowler		
	Comparison (Old 2025 / New 2025)					
	Acc	RMSE	SMAPE	Acc	RMSE	SMAPE
Gemini	57 / 60	9.1 / 5.8	16 / 7	60 / 65	3.4 / 1.9	17 / 6
2.5 Flash	+3	-3.24	-9	+5	-1.51	-11
Llama	37 / 36	20.3 / 20.2	37 / 33	36 / 45	8.6 / 11.0	38 / 25
(70b)	-1	-0.14	-4	+9	+2.48	-13
Qwen	31 / 32	20.5 / 19.8	50 / 42	38 / 44	8.2 / 11.7	39 / 27
(72b)	+1	-0.75	-8	+6	+3.57	-12
GPT	52 / 40	13.5 / 12.7	26 / 20	43 / 49	8.7 / 6.9	37 / 18
4.1	-12	-0.81	-6	+6	-1.80	-19

Table 7: Comparison of model performance on pre-2025 and post-2025 cricket samples.

As presented in Table 7, most models show gains in performance on the new cricket samples in the full summarization setting. Notably, GPT-4.1 shows a steep drop in accuracy for Batsman tables, signifying the intertwining of reasoning with pre-trained bias for specific entities.

	Basketball		
	Comparison (2021–2022 / 2025)		
	Acc	RMSE	SMAPE
Gemini	58 / 66	1.9 / 1.8	26 / 18
2.5 Flash	+7	-0.14	-8
Llama	29 / 30	4.3 / 3.4	63 / 62
(70b)	+1	-0.91	-1
Qwen	31 / 27	2.7 / 2.8	65 / 69
(72b)	-4	+0.06	+4
GPT	47 / 40	1.8 / 2.2	37 / 44
4.1	-7	+0.36	+7

Table 8: Comparison of model performance on basketball samples from 2021–2022 and 2025.

Table 8 shows that chronological effects in basketball are mixed rather than uniformly favoring the older **2021–2022** samples over the newer **2025** ones. Gemini improves substantially on 2025 games, with gains in both cell accuracy and numeric fidelity. Llama remains broadly stable while lowering RMSE on newer samples. In contrast, GPT-4.1 and Qwen both decline on 2025 data, showing drops in accuracy together with higher RMSE and SMAPE. This complements the cricket analysis by showing that temporal sensitivity is model-dependent and interacts with the structure of the sport, rather than reducing to a single monotonic trend.

A.2 Masked and Unmasked Commentary

	Batsman			Bowler		
	Comparison (Masked / Unmasked)					
	Acc	RMSE	SMAPE	Acc	RMSE	SMAPE
Gemini	49 / 86	13.82 / 4.26	22 / 4	43 / 55	7.83 / 4.93	28 / 15
2.5 Flash	+37	-9.56	-18	+12	-2.90	-13
Qwen	34 / 80	19.19 / 11.10	40 / 12	38 / 37	12.23 / 23.27	35 / 39
(72b)	+46	-8.09	-28	-1	+11.04	+4
Llama	39 / 85	19.76 / 6.71	32 / 7	40 / 42	11.95 / 18.14	29 / 30
(70b)	+46	-13.05	-25	+2	+6.19	+1
GPT	61 / 89	9.82 / 4.68	14 / 4	51 / 52	7.46 / 7.28	23 / 22
4.1	+28	-5.14	-10	+1	-0.18	-1

Table 9: Analysis of model performance on masked and unmasked cricket commentary.

Table 9 presents the comparison between masked (removing batsman summary cues) and unmasked commentaries. All models show substantial improvement in performance, validating that summary cues reduce this task from reasoning to simple extraction. Notably, for Gemini-2.5-Flash, performance on bowler tables also improves, with a significant reduction in error rates.

B Error Analysis

B.0.1 Error Analysis in Entity Entanglement

Under the **Entity-Entangled** condition, models do not make random entity-level errors; instead, they systematically confuse *roles*. Table 12 reports the number of extra rows, missing rows, and cross-role entity overlap between the batsman and bowler scorecards for different models and perturbations in the cricket domain.

We quantify this phenomenon using entity entanglement metrics. **Batsman Ent%** measures the proportion of entities in the predicted batsman scorecard that appear in the ground-truth bowler scorecard, while **Bowler Ent%** measures the proportion of entities in the predicted bowler scorecard that appear in the ground-truth batsman scorecard. Both metrics are computed using the Jaccard overlap, $\text{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|}$.

Across all models, we observe consistently high cross-role overlap under entity-entangled perturbations, indicating a failure to preserve role boundaries. We illustrate this behavior with a concrete example below in Table 10 and 11

Model Predictions under Entity Entanglement

Analysis In the ground truth, batsmen and bowlers form two strictly disjoint entity sets. Under entity entanglement, however, the model fails to

Batsman (GT)	Bowler (GT)
Bagai	Benn
Bhatti	Bernard
Cheema	Rampaul
Chohan	Sammy
Dhaniram	
Limbada	
Patel	
Surkari	
TC Bastiampillai	

Table 10: Ground-truth entities with strictly disjoint batting and bowling roles.

Bowler Pred (Entangled)	Batsman Pred (Entangled)
Rampaul	TC Bastiampillai
TC Bastiampillai	Rampaul
Cheema	Cheema
Bernard	Bernard
Surkari	Surkari
Limbada	Bagai
Benn	Limbada
Sammy	Benn
Patel	Sammy
Bhatti	Dhaniram
Dhaniram	Patel
Chohan	Bhatti
Chohan	

Table 11: Predicted scorecards under entity entanglement, showing cross-role mixing.

maintain these role distinctions and assigns entities interchangeably across roles.

Specifically, the predicted bowler scorecard contains eight entities that belong to the ground-truth batsman set (*Bhatti, Cheema, Chohan, Dhaniram, Limbada, Patel, Surkari, TC Bastiampillai*), yielding a **61.5%** Jaccard overlap with the batsman ground truth.

Conversely, the predicted batsman scorecard includes four ground-truth bowlers (*Benn, Bernard, Rampaul, Sammy*), resulting in a **30.7%** overlap with the bowler ground truth.

This example demonstrates that entity entanglement is fundamentally a *role-confusion failure mode*: the model does not merely misidentify entities, but systematically swaps their semantic roles when contextual signals are entangled.

B.0.2 Missing vs Extra Rows

Tables 12 and 13 summarize the number of missing and spurious entities produced by different models in the generated summary tables. In the original cricket setting, Gemini exhibits the highest number of missing entities relative to other models, but its performance improves when perturbed commentaries are provided. In contrast, the remaining models remain relatively stable in the cricket domain, showing limited variation in missing or extra entities under perturbations. The missing and extra rows percentages were checked against a range of name-matching thresholds from 0.5 to 0.1 making sure that the missing and extra entities are not comprised of entities that failed the name-match test.

The observed instability under the entity-entangled condition can be attributed to confusion between batsmen and bowlers, where entities are systematically interchanged across roles. In the basketball domain, which contains substantially more entities than cricket, all models exhibit a higher number of missing entities under perturbed settings. Models consistently miss more entities in basketball than in cricket, suggesting that an increase in the number of entities adversely impacts models' ability to reliably track and preserve entity representations.

C Temporal Robustness

In this section, we analyze the stability of model performance over time, specifically investigating how context length and the presence of explicit summary cues influence the ability to reconstruct scorecards across an innings.

C.1 Impact of Summary Cues on Temporal consistency

We have discussed how presence of cues or summaries affect models' performance in Table 9, Figure 3 reveals a distinct temporal trend when summary cues are present.

As shown in Figure 3(a), batsman cell accuracy rises steadily across overs. This suggests that as wickets fall, explicit mentions of dismissals allow models to extract data directly rather than inferring it. This indicates that high performance in the presence of summaries is largely extractive rather than reasoning-based.

C.2 Reasoning Without Cues: Temporal Trends

To isolate genuine temporal reasoning capabilities, we evaluated models without summary cues. Figure 4 illustrates the accuracy trends over the course of an innings in this harder setting.

	Batsman			Bowler		
	Entity Subs. (OOD) (Original / Perturbed)					
Model	Extra%	Miss%	Ent%	Extra%	Miss%	Ent%
Gemini	0 / 0	8 / 1	0 / 0	0 / 0	5 / 0	0 / 0
Delta	+0	-7	+0	+0	-5	+0
Qwen (72b)	0 / 1	3 / 3	0 / 0	0 / 0	2 / 1	0 / 0
Delta	+1	+0	+0	+0	-1	+0
Llama (70b)	0 / 0	1 / 1	0 / 0	0 / 0	0 / 0	0 / 0
Delta	+0	+0	+0	-0	+0	+0
GPT-4.1	1 / 1	1 / 1	0 / 0	0 / 0	0 / 0	0 / 0
Delta	+0	+0	+0	+0	-0	+0
	Entity Entangled (Original / Perturbed)					
Model	Extra%	Miss%	Ent%	Extra%	Miss%	Ent%
Gemini	0 / 6	8 / 1	0 / 6	0 / 6	5 / 1	0 / 5
Delta	+6	-7	+6	+6	-4	+5
Qwen (72b)	0 / 12	3 / 8	0 / 10	0 / 7	2 / 10	0 / 6
Delta	+12	+5	+10	+7	+8	+6
Llama (70b)	0 / 6	1 / 3	0 / 5	0 / 5	0 / 1	0 / 4
Delta	+6	+2	+5	+5	+1	+4
GPT-4.1	1 / 6	1 / 1	0 / 4	0 / 3	0 / 1	0 / 2
Delta	+5	+0	+4	+3	+1	+2
	Anonymized (Original / Perturbed)					
Model	Extra%	Miss%	Ent%	Extra%	Miss%	Ent%
Gemini	0 / 1	8 / 2	0 / 0	0 / 0	5 / 1	0 / 0
Delta	+1	-6	+0	+0	-4	+0
Qwen (72b)	0 / 1	3 / 6	0 / 0	0 / 0	2 / 3	0 / 0
Delta	-1	-3	+0	+0	-1	+0
Llama (70b)	0 / 2	1 / 3	0 / 0	0 / 5	1 / 1	0 / 0
Delta	-2	-2	+0	-5	+0	+0
GPT-4.1	1 / 1	1 / 1	0 / 0	0 / 0	0 / 0	0 / 0
Delta	+0	+0	+0	-0	-0	+0

Table 12: Comparison of entity tracking accuracy under perturbations for different models on the cricket dataset. Extra (%) denotes the proportion of spurious entities generated that are not present in the ground truth, Miss (%) denotes the proportion of ground-truth entities omitted by the model, and Ent (%) measures cross-role entity entanglement, defined as the percentage of predicted entities appearing in the opposing ground-truth role (batsman vs. bowler).

Models	Error Rates	
	Counterfactual (Original / Perturbed)	
	Extra%	Miss%
Gemini-2.5-Flash	0 / 41	1 / 7
Delta	+41	+6
Qwen-2.5-72B-Instruct	1 / 33	3 / 15
Delta	+32	+12
Llama-3.3-70B-Instruct	1 / 40	1 / 9
Delta	+39	+8
GPT-4.1	1 / 44	1 / 3
Delta	+43	+2
	Anonymized (Original / Perturbed)	
	Extra%	Miss%
Gemini-2.5-Flash	0 / 50	1 / 19
Delta	+50	+18
Qwen-2.5-72B-Instruct	1 / 49	3 / 27
Delta	+48	+24
Llama-3.3-70B-Instruct	1 / 39	1 / 19
Delta	+38	+18
GPT-4.1	1 / 68	1 / 45
Delta	+67	+44

Table 13: Comparison of entity tracking accuracy under perturbations for different models on the basketball dataset. Extra (%) denotes the proportion of spurious entities generated that are not present in the ground truth, and Miss (%) denotes the proportion of ground-truth entities omitted by the model.

Model	AUC (Batsman)	AUC (Bowler)
Gemini-2.5-Flash	**3.25**	**3.35**
Llama-3.3-70B-Instruct	1.87	1.57
Qwen-2.5-72B-Instruct	1.59	1.83
GPT-4o-mini	1.28	1.41

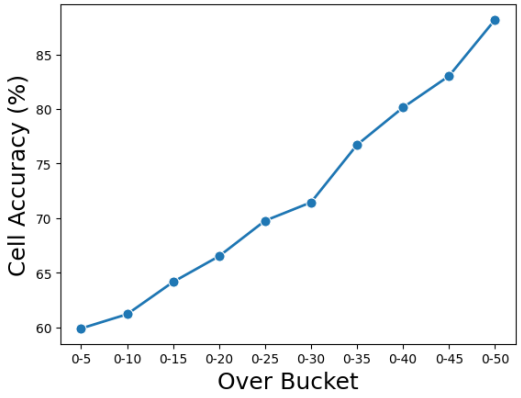
Table 14: AUC scores computed from temporal accuracy curves. Higher values indicate stronger consistency in maintaining game state across long contexts.

Unlike the cue-assisted setting, batsman accuracy without summaries declines monotonically with increasing input length (Figure 4). This confirms that without explicit state-resets (summaries), models struggle to maintain the hidden state of the game over long contexts. The decline stabilizes between overs 20–30, a "middle-overs" phase characterized by fewer events and reduced contextual drift.

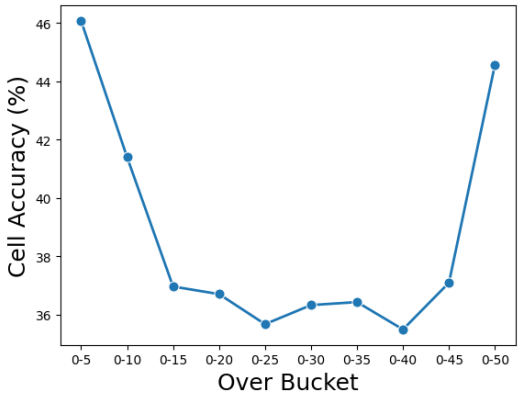
Bowler accuracy curves are notably flatter. While overall accuracy is lower (consistent with the commentary's batsman-centric bias), the lack of sharp fluctuations suggests that bowling reconstruction relies more on aggregate reasoning than on specific lexical triggers.

C.3 Quantifying Consistency: AUC Analysis

To quantify these trends, we computed the Area Under the Curve (AUC) by averaging cell, row, and column accuracies across all overs. This metric serves as a proxy for a model's temporal stability. As shown in Table 14, **Gemini-2.5-Flash** demonstrates superior temporal robustness, achieving the highest AUC for both batsman and bowler tables. Notably, **Qwen-2.5-72B-Instruct** outperforms **Llama-3.3-70B-Instruct** on the bowler table, suggesting it may possess stronger aggregate reasoning capabilities despite being less effective at instantaneous extraction. The data confirms that while larger models generally maintain state better, the specific architecture (e.g., Gemini's long-



(a) Batsman cell accuracy (Gemini-2.5-Flash).



(b) Bowler cell accuracy (Gemini-2.5-Flash).

Figure 3: Temporal cell accuracy trends **with summary cues** for (a) batsman and (b) bowler tables. Note the rising trend in batsman accuracy.

context optimization) plays a critical role in temporal robustness.

D Implementation Details and Practical Considerations

To ensure deterministic behavior, we use a fixed temperature of 0.1 and top-p of 0.1. Llama and Qwen models are run locally on two nodes of H200 clusters, while Gemini and GPT models are accessed via Google's and OpenAI's official API endpoints, respectively. Beyond aggregate performance, several baselines exhibited reliability issues that materially affected feasibility, throughput, and compute cost. We summarize these behaviors to contextualize the reported results and clarify the operational trade-offs of different modeling strategies.

D.1 Reliability and Regeneration Overhead

Table 16 complements the main-paper performance results by explicitly measuring the regeneration burden behind them. Whereas Table 1 estimates the *upper-bound accuracy* attained after allowing methods to retry until a structurally valid output is produced, the table below measures how expensive that process is in practice. We evaluate a representative subset of 100 samples per sport for Gemini-2.5-Flash and Llama-3.3-70B-Instruct, allow all methods to regenerate until valid outputs are obtained, log every pass (successful and failed), and report: (i) first-pass failure rate, (ii) total passes required to achieve complete coverage, and (iii) average input/output token usage aggregated over all runs and divided by sample size.

Reliability Challenges For Text-Tuple-Table, we observed frequent degenerate repetitive patterns in which models repeatedly emitted identical tuples hundreds or thousands of times without termination. Such outputs were unparsable and required re-generation. These failures were most common when using flat tuple formats of the form `(entity, attribute, value)`, which proved less stable for autoregressive generation than structured JSON outputs.

Llama-3.3-70B-Instruct exhibited a systematic cumulative counting behavior on high-frequency attributes such as `Balls_Faced` and `Balls`, generating running totals (e.g., 1,2,3,4,...) rather than incremental updates. This led to extreme numerical errors, with RMSE values exceeding 100 on many samples. Correct outputs were obtained only after repeated regeneration, substantially increasing compute requirements.

Due to these reliability issues, multiple passes and iterative engineering of the prompts were required to successfully process the full dataset. Under Text-Tuple-Table, GPT-4.1 requires 3 passes (36/1362, or 2.6%, failure cases in the first pass) to obtain valid outputs for all samples, with roughly 20% of samples needing regeneration. Llama-3.3-70B-Instruct required up to 15 passes (648/1362, or 47.6%, failure cases in the first pass) to achieve complete coverage, significantly increasing wall-clock time despite its lower per-token cost when run locally. Gemini-2.5-Flash was the most effective model, with only 2% failure cases, requiring only two retries.

Table 15 details the computational footprint across all baselines, highlighting the significant variance in resource consumption required by structured reasoning frameworks. Standard CoT remains the most economical approach, serving as

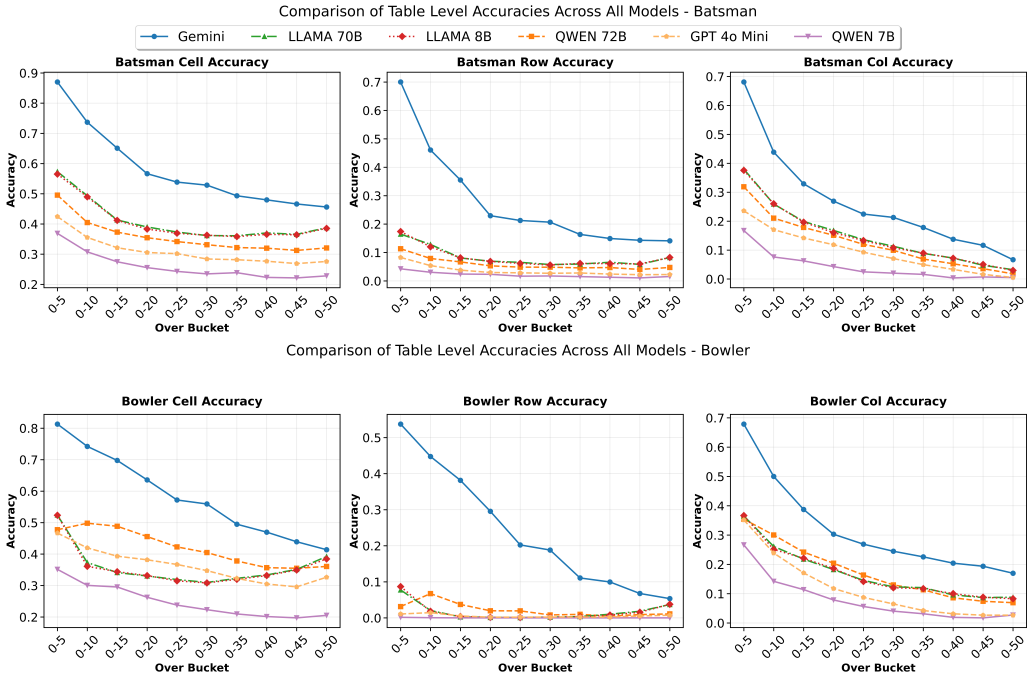


Figure 4: Temporal trends of accuracy on cricket score tables.

	Cricket			Basketball		
Baseline Method	**API Calls**	**Input Tok.**	**Output Tok.**	**API Calls**	**Input Tok.**	**Output Tok.**
CoT	1	7,136	358.6	1	5,116	826
Intermediate Sum. (L2M, n=2)	2	7,975.6	521.2	3	6,852.7	1,577.3
Intermediate Sum. (L2M, n=4)	4	9,461.2	791.8	5	9,223.4	2,546.1
Intermediate Sum. (L2M, n=8)	8	12,411.3	1,301.2	9	13,189.4	3,947.5
EntityCoT	16.1	28,713.1	1,121.4	24.3	17,657.1	2,547.5
T3 with Rule Integration	3	9,756	**7,515.5**	4	8,138	**7,813.1**

Table 15: Resource usage comparison across baselines for cricket and basketball (on successful samples).

the efficiency baseline with a single API call and minimal token usage (5–7k input tokens). In contrast, EntityCoT exhibits the highest computational demand, requiring an average of 16.1 to 24.3 API calls per instance, an increase of over 20× compared to CoT, along with the highest input token load (28.7k for Cricket). Meanwhile, T3 presents a unique profile: while its API call overhead is moderate (3–4 calls), it generates significantly higher output volumes (7.5k tokens), reflecting its comprehensive table-generation process. The Intermediate Summarisation (L2M) methods demonstrate a linear scaling in cost, where increasing n from 2 to 8 proportionally raises API calls and token usage without reaching the resource intensity of EntityCoT.

Model-Specific Feasibility Constraints. Qwen models were not evaluated on the most complex baselines due to extremely low throughput and persistent instability in producing structured outputs under these settings. Accordingly, tuple-based and entity-centric baselines are reported only for models capable of reliably emitting well-formed structured representations. Overall, these observations highlight that high average performance does not necessarily imply practical deployability, and that reliability, retry behavior, and structured output stability are critical factors for long-context table generation.

E Results Tables

We have done our analysis on two sets of data: ODI cricket (50 over matches) and NBA basketball. The nature of data differs not only in terms of sport but also in other aspects like recency and temporal span—the basketball data spans 2021–2022

Method	Cricket				Basketball			
	Fail. Rate	Passes	Input Tok.	Output Tok.	Fail. Rate	Passes	Input Tok.	Output Tok.
Llama-3.3-70B-Instruct								
CoT	2%	2	6,579	348	0%	1	6,633	794
Divide and Generate ($n = 2$)	0%	1	7,579	514	3%	2	7,384	1,389
Divide and Generate ($n = 4$)	0%	1	9,065	786	2%	2	10,181	2,948
Divide and Generate ($n = 8$)	0%	1	12,036	1,300	7%	4	12,980	4,384
Entity CoT	0%	1	25,165	954	0%	1	18,232	2,380
T3	47%	7	16,938	12,665	5%	3	8,678	8,878
Gemini-2.5-Flash								
CoT	2%	2	6,705	330	5%	2	5,307	630
Divide and Generate ($n = 2$)	1%	2	7,452	479	2%	2	6,035	1,165
Divide and Generate ($n = 4$)	3%	2	9,189	763	0%	1	7,411	2,069
Divide and Generate ($n = 8$)	0%	1	11,934	1,231	3%	2	10,553	3,542
Entity CoT	0%	1	20,311	999	8%	3	16,926	2,418
T3	1%	2	9,447	7,466	0%	1	8,370	7,858

Table 16: Reliability and regeneration overhead on a representative subset of 100 samples per sport. All methods are allowed to regenerate until structurally valid outputs are obtained; token costs include both successful and failed runs and are averaged per sample.

and 2025, in contrast to ODI cricket which spans from 2006–2025. Additionally, the datasets differ in their structural characteristics: basketball features continuous fluid play with frequent position switches, while cricket has discrete, turn-based ball-by-ball commentary with fixed role distinctions between batsmen and bowlers.

F Ground Truth Generation

F.1 Structuring Cricket Commentary from CricInfo

We convert every ball bowled into a structure using regexes to extract the batsman and bowler entities which always have a pattern. We also extract the runs that comes off that bowl to keep tracking the numerical attributed linked to bowler and batsman. The pseudocode for the same has been explained below.

```
Input: raw_commentary_text

1) Segment into balls
   - Find over markers with regex: r"\b
     ([0-4]?[0-9]\.[1-6])\b"
   - For each adjacent pair of matches, slice
     text between them → ball_chunk

2) Parse each ball_chunk
   - Split by lines → L
   - over := float(first space-token in
     ball_chunk)  // from L joined/split by
     spaces
   - run_token := normalize(L[1]); map "•"→"0"
   - Extract bowler,batter from L[2] using:
       regex: r"^(?P<bowler>.+?) to (?P<batter
     >.+?)," → groups["bowler"], groups["batter
     "]
   - commentary := concatenate L[2:] until any
     stop condition:
       * contains "CRR:" or "end of over"
       * contains time token via regex: r"\b\d
     {2}(?:am|pm)\b" (case-insensitive)
       * contains " SR: "
     (Stop at first hit; keep preceding text.)

3) Derive per-ball features
   - Team runs increment:
       if run_token starts with digit → int(
     run_token[0]); else 0
   - Batter runs:
       if regex r"^\d+$" → int(run_token)
       elif regex r"^\d+nb$" → int(first digit)
     - 1
       else 0
   - Batter 4/6:
       four := 1 if (run_token == "4" or regex r
     "^4(?!lb)") else 0
       six  := 1 if (run_token == "6" or regex r
     "^6(?!lb)") else 0
   - Batter ball faced:
       1 if (regex r"^\d+$" or run_token == "W"
     or (no 'w' in run_token)) else 0
   - Bowler legal ball:
       1 if (regex r"^\d+$" or (no 'w' and no '
     nb' in run_token)) else 0
   - Bowler runs conceded:
       if regex r"^\d+$" → int(run_token)
       elif regex r"^\d+(?:w|nb)$" → int(first
     digit)
       else 0
   - Wicket (bowler credit):
       if run_token == "W" AND commentary
     contains any dismissal keyword:
         [" st "," c "," lbw "," hit wicket ","
     b "] → wicket = 1
       else 0

4) Continuity enforcement
   - Compute ball index: idx(over) = floor(over)
     *6 + int(10*(over - floor(over)))
   - If previous run was wide/nb (regex r"^\d+(?:
     w|nb)$"), expect same idx; else expect idx
     +1.
   - Skip chunk if expectation not met.

5) Collect row
   - Append: {raw_text, over, run_token,
     team_runs_inc, commentary,
              bowler, batter, batter_runs, four,
      six, batter_balls,
```

```
            bowler_balls, bowler_runs, wicket,
    dismissal_phrase}

6) Build DataFrame from all rows
```

F.2 Commentary Ground Truth: Cricket

A sample cricket commentary from our dataset has been shown along with the ground truth that was calculated using the pipeline explained in section F.

```
Muzarabani to Tamim, no run fullish ball on
    middle and leg, clipped to square leg

Muzarabani to Tamim, no run full and straighter
    on off this time. Driven down the ground
    with the full face of the bat

Muzarabani to Tamim, no run past the outside
    edge to the batter. Fullish length in the
    off-stump channel, as Tamim presents his bat
     playing in the line and the ball shoots of
    the the pitch

Muzarabani to Tamim, 1 wide good length ball
    pitched outside leg, as he shoulders arms.
    Wide called

Muzarabani to Tamim, no run fuller on middle and
     off. He goes across the stumps to clip that
     to square leg

Muzarabani to Tamim, no run beaten on the
    outside edge. Good length delivery pitched
    on middle and leg, and angles away as he
    tries to dab at that with an open face

Muzarabani to Tamim, no run good length again,
    this time on off. Good bounce for Muzarabani
     again, as Tamim blocks that off the front
    foot close to his body

Chatara to Litton Das, no run full and wide of
    off to perhaps draw the drive, and angles
    away further as Liton shoulders arms

Chatara to Litton Das, no run good length ball
    on middle and off. Defended off the back
    foot to cover Two slips in place for Chatara

Chatara to Litton Das, no run full ball again
    and closer to off. This was around fifth
    stump, and Liton was happy to let that go
    again

Chatara to Litton Das, no run full length just
    outside off. Liton takes a good stride
    forward to tap that to cover

Chatara to Litton Das, no run fullish length and
     slightly wide of off. Straightens after
    pitching after Liton leaves it alone

Chatara to Litton Das, no run another full
    delivery just outside off. Defended to cover
     off the front foot

Muzarabani to Tamim, no run full length just
    outside off, and tapped to cover with an
    open face

Muzarabani to Tamim, no run pulls his length
    back a touch just outside off. Driven toward
     mid-on with the full face of the bat

Muzarabani to Tamim, FOUR runs slashed on the up
     through the covers and with a crackling
    sound. Good length ball outside off, and
    Tamim takes a solid foot forward to hit that
     aerially and holds the pose after that too

Muzarabani to Tamim, no run good length on
    middle and leg, clipped to square leg off
    the back foot

Muzarabani to Tamim, FOUR runs glorious drive
    between mid-off and cover. Presents the full
     face of the bat to this full delivery
    outside off, and gets the second boundary of
     the over

Muzarabani to Tamim, no run fullish length
    outside off, and Tamim plays within the line
     as the ball passes his outside edge

Chatara to Litton Das, no run beats him on the
    outside edge. Fullish, nagging length in the
     channel with the ball angling away as he
    tries to defend at that

Chatara to Litton Das, no run full ball on off
    again, and defended to point this time

Chatara to Litton Das, no run goes full but much
     closer to off stump. Driven to mid-off with
     the full face of the bat

Chatara to Litton Das, 1 wide fullish ball
    pitching on middle and leg, and angling down
     leg further. Liton shoulders arms, and it's
     called a wide

Chatara to Litton Das, no run another fullish
    ball, and straighter as well. Pushed off the
     front foot to cover

Chatara to Litton Das, FOUR runs flicked off his
     pads to deep square leg, as the bowler goes
     full on leg this time. Just clips that to
    the right of the square leg umpire and beats
     fine leg to his left

Chatara to Litton Das, no run driven off the
    full face to point, as Chatara bowls this on
     a good length and slightly wide of off

Muzarabani to Tamim, no run good carry for
    Muzarabani off a full length in the off-
    stump channel. Tamim shoulders arms to that,
     and the keeper collects that really high

Muzarabani to Tamim, 1 run pushes this off the
    outside edge with his front leg in the air
    as the bowler goes full and very close to
    off. Third man rushes to his right to dive
    and keep that to a single

Muzarabani to Litton Das, no run good length
```

```
    delivery on off, and angling in as he fends
    that to the on side off his chest

Muzarabani to Litton Das, no run full and on off,
     as Liton gets a leading edge back to the
    bowler while trying to flick that to the on
    side

Muzarabani to Litton Das, no run another full
    ball on off, and Liton is right forward to
    block that back

Muzarabani to Litton Das, no run the ball just
    evades his off stump as he defends this good
     length ball close to himself. It was
    angling in after pitching outside off, and
    Liton played that late with the ball
    escaping his off stump with a few bounces
    away
```

Tables 17 and 18 illustrate example ground-truth scorecards derived from a five-over cricket commentary segment. The **bowler table** reports the number of balls delivered, total runs conceded, wickets taken, overs bowled, and maiden overs for each bowler. The **batsman table** summarizes runs scored, balls faced, boundaries (fours and sixes), and strike rate (runs per 100 balls faced) for each batsman. These columns reflect standard cricket statistics and are deterministically aggregated from ball-level events.

Bowler	Balls	Runs	Wickets	Overs	Maidens
Chatara	12	5	0	2.0	1
Muzarabani	18	10	0	3.0	0

Table 17: Bowling summary.

Batsman	**Runs**	**Balls_Faced**	**Fours**	**Sixes**	**S/R**
Litton Das	4.0	16	1	0	25.0
Tamim	9.0	14	2	0	64.29

Table 18: Batsman summary.

F.3 Structuring Basketball Commentary

The flow below explains how Basketball commentaries are collected from ESPN.

```
Input: ESPN_play_by_play_pages

1) Scrape play-by-play data
   - Initialize thread pool for concurrent
     Selenium sessions
   - For each game URL:
       * Launch headless browser
       * Navigate to ESPN Deportes play-by-play
     page
       * Scroll/load dynamic content
       * Extract event elements (quarter, clock,
      description, score)
       * Handle timeouts and retry failures
   - Aggregate all events into raw JSON
     structure keyed by game_id

2) Parse raw JSON into structured commentary
   - For each game_id in JSON:
       * Group events by quarter
       * Within each quarter, sort by game clock
      (descending)
       * For each event:
           - Extract timestamp, event text,
     score snapshot
       * Concatenate events into chronological
     narrative blocks
   - Write one text file per game with quarter
     headers and timestamps

3) Clean commentary text
   - Iterate through generated text files
   - Apply regex filters to remove score
     identifiers:
       * Remove patterns like r"\(\d+-\d+\)"
       * Remove inline score tokens (e.g., "LAL
     102-98")
   - Normalize whitespace and line breaks
   - Preserve only descriptive play-by-play
     language

4) Output refined corpus
   - Save cleaned files into a parallel
     directory structure
   - Ensure one narrative-only text file per
     game
   - Output optimized textual dataset for LLM
     ingestion

Output: clean_narrative_commentary_corpus
```

F.4 Commentary Ground Truth: Basketball

A sample basketball commentary from our dataset has been shown along with the ground truth that was calculated using the pipeline explained in section F.3.

```
Nic Claxton vs. Brook Lopez (Giannis
    Antetokounmpo gains possession)
Grayson Allen misses 27-foot three point jumper
Kevin Durant defensive rebound
Giannis Antetokounmpo shooting foul
Nic Claxton misses free throw 1 of 2
Nets offensive team rebound
Nic Claxton misses free throw 2 of 2
Giannis Antetokounmpo defensive rebound
Giannis Antetokounmpo misses three point pullup
    jump shot
Brook Lopez offensive rebound
Giannis Antetokounmpo misses pullup jump shot
Blake Griffin defensive rebound
Nic Claxton makes alley oop dunk shot (James
    Harden assists)
Giannis Antetokounmpo misses 11-foot hook shot
Giannis Antetokounmpo offensive rebound
Giannis Antetokounmpo makes two point shot
James Harden misses 28-foot step back jumpshot
Giannis Antetokounmpo defensive rebound
Brook Lopez makes 28-foot three point jumper (
    Giannis Antetokounmpo assists)
Kevin Durant makes 13-foot two point shot (James
```

```
    Harden assists)
Giannis Antetokounmpo misses layup
Giannis Antetokounmpo offensive rebound
James Harden blocks Giannis Antetokounmpo's two
    point shot
```

Player	PTS	REB	AST	STL	BLK	TO	FGM	FGA	3PM	3PA
Brooklyn Nets										
N. Claxton	2	0	0	0	0	0	1	1	0	0
K. Durant	2	1	0	0	0	0	1	1	0	0
B. Griffin	0	1	0	0	0	0	0	0	0	0
J. Harden	0	0	2	0	1	0	0	1	0	1
Milwaukee Bucks										
Giannis A.	2	4	1	0	0	0	1	6	0	1
B. Lopez	3	1	0	0	0	0	1	1	1	1
G. Allen	0	0	0	0	0	0	0	1	0	1

Table 19: Q1 summary (compact view).

G Prompts

G.1 Chain of Thought (COT)

Prompt for Table Generation: Cricket

The prompt used for the generating the Batsman and Bowler tables using Zero-Shot (ZS) Chain of Thought (COT) has been shown below. We use ZS COT as it has been shown in (Deng et al., 2024a) that models improve in ZS setting after applying COT. It has also been shown by (Deng et al., 2024a) that few-shot learning does not improve performance and ZS performs far better than fine-tuning setting.

Prompt Card.

System message

System Prompt

Role: You are an expert in building statistical summary tables from live commentary text. Return ONLY the two JSON objects according to the rules given below - no prose, no code, no markdown.

Output Rules:
- Two lines total: line 1 = batsman JSON; line 2 = bowler JSON.
- Use EXACT key casing: **batsman**, **Runs**, **Balls_Faced**, **Fours**, **Sixes**, **S/R**, **dismissal**; **bowler**, **Balls**, **Runs_Given**, **Wickets**, **Overs**, **Maidens**.
- Arrays align by index (e.g., batsman[i] ↔ Runs[i] ↔ Balls_Faced[i]).
- Types: integers for count fields; **S/R** is a float with two decimals; **Overs** is "O.B" string (e.g., "3.4").
- Unknown values: use 0 (numbers) or "" (strings). Never output null/NaN.

Cricket Rules:

[Batsman]
- **Runs**: credit only off-the-bat (exclude byes/leg-byes). For no-balls, bat runs are credited separately; the +1 nb is NOT bat runs.
- **Balls_Faced**: increment for every delivery except wides (no-balls DO count as a ball faced).
- **Fours/Sixes**: set only for off-the-bat boundaries; exclude b/lb boundaries.
- **S/R**: (Runs / Balls_Faced) * 100, rounded to 2 decimals.
- **dismissal**: "" if not out; else short phrase such as "c Fielder b Bowler", "lbw", "run out", "st", "b Bowler", "hit wicket".

[Bowler]
- **Balls**: count only legal deliveries (wides/no-balls do NOT add to Balls).
- **Runs_Given**: includes all conceded (incl. wides, no-balls, byes/leg-byes).
- **Wickets**: credit only bowler-attributable dismissals (b, c off bowler, lbw, st off bowler, hit wicket). Do NOT credit run out.
- **Overs**: floor(Balls/6) "." (Balls % 6).
- **Maidens**: count overs with zero **Runs_Given** in that over (0 if not inferable).

Validation:
- Output must be valid JSON with no trailing commas and no extra lines.
- Arrays must be equal length per table.

User message.

User Message: Data Injection & CoT Trigger

Instruction: You are about to be given a cricket commentary. Extract the final batsman and bowler scorecards by aggregating across the commentary. Return ONLY the two JSON objects as specified.

Commentary:

```
{commentary}
```

Output format (copy keys verbatim; replace placeholders with values):

```
{
   "batsman":[batsman1, batsman2, ...],
   "Runs":[runs1, runs2, ...],
   "Balls_Faced":[balls1, balls2, ...],
   "Fours":[fours1, fours2, ...],
   "Sixes":[sixes1, sixes2, ...],
   "S/R":[sr1, sr2, ...],
   "dismissal":[dis1, dis2, ...]
}
{
   "bowler":[bowler1, bowler2, ...],
   "Balls":[balls_b1, balls_b2, ...],
   "Runs_Given":[runs_g1, runs_g2, ...],
   "Wickets":[wkts1, wkts2, ...],
   "Overs":[overs1, overs2, ...],
```

```
  "Maidens":[m1, m2, ...]
}
```

Magic Phrase: Let's think step by step.

Prompt for Table Generation: Basketball

Prompt Card.

System message

System Prompt: Basketball (CoT)

Role: You are an expert in building statistical summary tables from live basketball play-by-play commentary text. Return ONLY one valid JSON object, following the rules below. No prose, no explanations, no markdown, no code blocks.

Output Rules:

- One line total: a single JSON object.
- Exact key casing (lowercase only): `player`, `pts`, `reb`, `ast`, `stl`, `blk`, `to`, `fg_made`, `fg_attempted`, `3pt_made`, `3pt_attempted`.
- Arrays must align by index (e.g., player[i] ↔ pts[i] ↔ reb[i] . . .).
- All stat values must be integers. Unknown/non-inferable values must be 0.
- Never output null, NaN, missing keys, or extra keys. Do not invent players.

Basketball Stat Definitions:

- `pts` (Points): +2 for made 2pt FG; +3 for made 3pt FG; +1 for made FT. (Do NOT award for misses/fouls).
- `fg_made`: +1 for any made FG (2pt or 3pt). Exclude FTs.
- `fg_attempted`: +1 for any FG attempt (made or missed). Exclude FTs.
- `3pt_made`: +1 for a made 3-point shot.
- `3pt_attempted`: +1 for any 3-point shot attempt (made or missed).
- `reb` (Rebounds): Count offensive and defensive. (+1 per explicit credit).
- `ast` (Assists): Credit ONLY when explicitly stated (e.g., "assisted by X"). Do not infer.
- `stl` (Steals): Credit ONLY on explicit terms (steal, strip, interception).
- `blk` (Blocks): Credit ONLY on explicit "blocked shot" statements.
- `to` (Turnovers): Credit ONLY when explicitly attributed to a player.

General Rules:

- If a player is mentioned but no stats are inferable, include player with all zeros.
- Aggregate stats across the entire commentary.
- Output must be valid JSON with no trailing commas and no extra lines.

User message.

User Message: Basketball Data Injection

Instruction: You are about to be given basketball commentary. Extract the final per-player boxscore by aggregating across the commentary. Return ONLY the single JSON object described above.

Commentary:

```
{commentary}
```

OUTPUT FORMAT (COPY EXACTLY):

```
{
  "player":[p1, p2, ...],
  "pts":[...],
  "reb":[...],
  "ast":[...],
  "stl":[...],
  "blk":[...],
  "to":[...],
  "fg_made":[...],
  "fg_attempted":[...],
  "3pt_made":[...],
  "3pt_attempted":[...]
}
```

G.2 Entity Chain of Thought (Entity-COT)

Entity-COT for Basketball

STEP 1: ENTITY EXTRACTION PROMPT

Role: You are an expert at extracting all player names from basketball play-by-play commentary.
Instruction: Given the following commentary, return a STRICTLY valid JSON object with exactly one field:

- `"players"`: a list of unique full player names (strings)

Rules:

- Return ONLY valid JSON, no prose.
- Use full names as they appear in commentary.
- Deduplicate and normalize spacing.

Commentary: {commentary}

STEP 2: SINGLE-PLAYER STAT GENERATION

System Prompt: You are an expert basketball statistician.
User Prompt: Given this game commentary, provide ONLY the final, fully-aggregated box score for player **"{player}"** as a JSON object with these keys: ["player", "pts", "reb", "ast", "stl", "blk", "to", "fg_made", "fg_attempted", "3pt_made", "3pt_attempted"]

Basketball Rules:

- `pts`: total points scored
- `reb`: total rebounds

- **ast**: total assists
- **stl**: total steals
- **blk**: total blocks
- **to**: total turnovers
- **fg_made**: total field goals made (includes 2pt and 3pt)
- **fg_attempted**: total field goal attempts
- **3pt_made**: total three-pointers made
- **3pt_attempted**: total three-point attempts

Constraint: Output only the JSON for this player. Do not invent data. Use 0 if not available.

Example Output:

```
{
  "player": "LeBron James",
  "pts": 28, "reb": 8, "ast": 7,
  "stl": 2, "blk": 1, "to": 3,
  "fg_made": 10, "fg_attempted": 20,
  "3pt_made": 3, "3pt_attempted": 8
}
```

Entity-COT for Cricket

STEP 1: ENTITY EXTRACTION PROMPT

Role: You are an expert at extracting all batsman and bowler names from cricket commentary.
Instruction: Given the following commentary, return a STRICTLY valid JSON object with exactly two fields:

- **"batsmen"**: a list of unique full batsman names (strings)
- **"bowlers"**: a list of unique full bowler names (strings)

Rules:

- Return ONLY valid JSON, no prose.
- Use full names as they appear in commentary.
- Deduplicate and normalize spacing.

Commentary: {commentary}

STEP 2: SINGLE-BATSMAN STAT GENERATION

Role: You are an expert cricket statistician.
Instruction: Given this match commentary, provide ONLY the final, fully-aggregated batting scorecard for batsman **"{batsman}"** as a JSON object with these keys: ["batsman", "Runs", "Balls_Faced", "Fours", "Sixes", "S/R", "dismissal"]

Cricket Rules:

- **Runs**: Credit only off-the-bat (exclude byes/leg-byes). For no-balls, bat runs are credited separately (+1 nb is NOT bat runs).
- **Balls_Faced**: Increment for every delivery except wides (no-balls DO count).
- **Fours/Sixes**: Off-the-bat boundaries only.
- **S/R**: (Runs/Balls_Faced) * 100, rounded to 2 decimals.
- **dismissal**: "" if not out; else phrase ("c Root b Anderson", "lbw", etc.).

Example Output:

```
{
  "batsman": "KL Rahul",
  "Runs": 37, "Balls_Faced": 24,
  "Fours": 2, "Sixes": 1,
  "S/R": 154.16,
  "dismissal": "c Root b Anderson"
}
```

STEP 3: SINGLE-BOWLER STAT GENERATION

Role: You are an expert cricket statistician.
Instruction: Given this match commentary, provide ONLY the final, fully-aggregated bowling scorecard for bowler **"{bowler}"** as a JSON object with these keys: ["bowler", "Balls", "Runs_Given", "Wickets", "Overs", "Maidens"]

Cricket Rules:

- **Balls**: Count only legal deliveries (wides/no-balls do NOT add).
- **Runs_Given**: All conceded (incl. wides, no-balls, byes/lbs).
- **Wickets**: Bowler-attributable only (b, c, lbw, st, hit wicket). No run outs.
- **Overs**: floor(Balls/6) "." (Balls % 6).
- **Maidens**: Overs with zero **Runs_Given**.

Example Output:

```
{
  "bowler": "Anderson",
  "Balls": 36, "Runs_Given": 55,
  "Wickets": 2, "Overs": "6.0",
  "Maidens": 1
}
```

G.3 Text-Tuple-Table (T^3)

T^3 Prompt for Cricket

SHARED COMPONENT: DOMAIN INSTRUCTION

(This text is injected into the {instruction} slot for both steps below)
Task: Summarise live commentary into two tables: Batsman Scorecard & Bowler Scorecard.

Cricket Logic Definition:

- **Runs (Batsman)**: Credit off-the-bat only.

No byes/leg-byes.
- **Balls_Faced**: Count all deliveries except wides. (No-balls count).
- **S/R**: $(Runs/Balls\_Faced) * 100$ (rounded to 2 decimals).
- **Runs_Given (Bowler)**: Count ALL conceded (wides, NBs, byes, leg-byes).
- **Maidens**: Count complete overs (6 legal balls) with zero **Runs_Given**.
- . . . [Remaining field definitions omitted for brevity]

STEP 1: EVENT-TO-TUPLE EXTRACTION

Prompt Template: According to the following instruction, please extract the relevant events and information in the form of tuples, structured as (Entity, Attribute, Value).

Extraction Guidelines:
- Format: (Entity, Attribute, Value)
- Atomic: Each tuple represents ONE specific stat update.
- Multiplicity: One commentary line → Multiple Tuples.

Few-Shot Examples (Enforced Output Style):

```
Input: "Muzarabani to Tamim, no run"
Output:
(Tamim, Balls_Faced, 1)
(Muzarabani, Balls, 1)
(Muzarabani, Runs_Given, 0)
Input: "Muzarabani to Tamim, FOUR runs"
Output:
(Tamim, Balls_Faced, 1)
(Tamim, Runs, 4)
(Tamim, Fours, 1)
(Muzarabani, Balls, 1)
(Muzarabani, Runs_Given, 4)
```

Input: <Text> {commentary} </Text>

STEP 2: SYMBOLIC AGGREGATION

Instruction: Aggregate these tuples into the final output tables.
- Group tuples by **Entity**.
- Sum all numeric **Attributes**.
- Apply derived calculations (e.g., Strike Rate).

Input Payload: <Tuples> {generated_tuples_from_step_1} </Tuples>

Required Output Format:
- Line 1: **{"batsman": [...], "Runs": [...], ...}**
- Line 2: **{"bowler": [...], "Balls": [...], ...}**

T^3 Prompt for Basketball

SHARED COMPONENT: DOMAIN INSTRUCTION

Task: Summarise live basketball commentary into a single player statistics table.

Basketball Logic Definition:
- **pts**: Total points (FG + FT).
- **fg_made/attempted**: Includes 2pt and 3pt.
- **3pt_made/attempted**: Specific subset of FGs.
- **reb**: Total rebounds (Offensive + Defensive).
- **ast, stl, blk, to**: Standard counting stats.

Required Output Structure:
- Single JSON object.
- Arrays align by index: player[i] ↔ pts[i] ↔ reb[i].

STEP 1: EVENT-TO-TUPLE EXTRACTION

Prompt Template: According to the following instruction, please extract the relevant events and information in the form of tuples, structured as (Entity, Attribute, Value).

Few-Shot Examples (Enforced Output Style):

```
Input: "LeBron James makes a 2-point field goal"
Output:
(LeBron James, pts, 2)
(LeBron James, fg_made, 1)
(LeBron James, fg_attempted, 1)
Input: "Stephen Curry makes a 3-pointer"
Output:
(Stephen Curry, pts, 3)
(Stephen Curry, fg_made, 1)
(Stephen Curry, fg_attempted, 1)
(Stephen Curry, 3pt_made, 1)
(Stephen Curry, 3pt_attempted, 1)
Input: "Kevin Durant grabs a rebound"
Output: (Kevin Durant, reb, 1)
```

Input: <Text> {commentary} </Text>

STEP 2: SYMBOLIC AGGREGATION

Instruction: Aggregate these tuples into the final output table.
- Group tuples by **Entity**.
- Sum all numeric **Attributes**.
- Return ONLY the final output in the exact format below.

Input Payload: <Tuples>

{generated_tuples_from_step_1} </Tuples>

Required Output Format:

```
{
  "player": [p1, p2, ...],
  "pts": [...],
  "reb": [...],
  "ast": [...],
  "stl": [...],
  "blk": [...],
  "to": [...],
  "fg_made": [...],
  "fg_attempted": [...],
  "3pt_made": [...],
  "3pt_attempted": [...]
}
```