TENOR: Tension-Enhanced Narrative Planning and Organization for Report Generation

Aliah Gie Fajardo Zabala[1], **Yi-Shin Chen**[1,2]
[1]Institute of Information Systems and Applications, [2]Department of Computer Science,
National Tsing Hua University, Hsinchu, Taiwan
Correspondence: yishin@gmail.com

Abstract

A final score records how a match ended, but its significance depends on how the match unfolded. Structured sports records contain scores and point outcomes, yet they do not directly indicate which phases and events help explain the result. We introduce TENOR, a Tension-Enhanced Narrative Planning and Organization framework for badminton table-to-text generation. TENOR adds an intermediate planning stage that models rally-level tension as outcome uncertainty, uses changes in the tension sequence to partition each set, selects focal events with supporting context, and guides segment-, set-, and match-level generation. We evaluate TENOR on 50 matches against in-context, reasoning-based, and table-oriented baselines. At the match level, the full TENOR system produces 35.12 supported relations, compared with 23.94 for the prompt-based baseline with the most supported relations despite a lower average output length, while maintaining 100.00 relation-grounded precision. It also achieves the highest PARENT recall and F1, the smallest precision–recall gaps in both Content Selection and PARENT, and the strongest coherence judgments across individual and ensemble LLM evaluators. Together, these results support TENOR's goal of producing grounded, coherently organized reports that explain not only the result but how it was reached.

1 Introduction

Sports matches are dynamic sequences of play, but structured datasets represent them as tables of rallies, score states, and outcomes. These records provide factual grounding for generation, but a surface reading can flatten a match into isolated events. For example, rallies at 4–4 and 19–19 both appear in the table as tied score states. To a human reader, however, their significance depends on where they occur in the developing set. At 4–4, the set is still taking shape. At 19–19, it is near resolution, and the next rally may change the direction of the set. Prior data-to-text and table-oriented methods use content planning, table operations, or decomposition to provide structure before generation (Wiseman et al., 2017; Puduppully et al., 2019; Wang et al., 2024; Chiang et al., 2025), but match progression may remain implicit.

This raises the question: can a system explain not only the result, but how it was reached? We address this question through an intermediate structure between record and report. We introduce **TENOR**, a Tension-Enhanced Narrative Planning and Organization framework for badminton table-to-text generation. TENOR treats a match as a layered account in which moments form phases, phases shape sets, and sets form the match. It models tension as outcome uncertainty, uses changes in tension to partition each set into phases, selects focal rallies with supporting context, and connects segment-, set-, and match-level accounts through hierarchical generation. The goal is therefore to produce grounded reports that make the path to the final result explicit through selected and organized match developments.

Across 50 matches, TENOR produces 35.12 supported relations, compared with 23.94 for Tree-of-Thought, the prompt-based baseline with the most supported relations. TENOR does so with shorter reports and 100.00 relation-grounded precision. It also achieves the highest PARENT recall and F1 and the strongest coherence judgments across individual and ensemble evaluators.

2 Related Work

2.1 Structured Support Before Generation

Data-to-text generation requires selecting relevant facts, ordering them coherently, and avoiding unsupported content (Wiseman et al., 2017). Explicit content planning addresses these decisions before surface realization (Puduppully et al., 2019).

Proceedings of the 19th International Natural Language Generation Conference: Long Papers, pages 972–991
October 17-21, 2026 ©2026 Association for Computational Linguistics

Prompting and reasoning methods offer other forms of support before generation. In-context learning supplies task instructions or demonstrations (Brown et al., 2020), while Chain-of-Thought and Tree-of-Thought introduce reasoning steps or explore alternative reasoning paths (Wei et al., 2022; Yao et al., 2023).

Table-oriented methods make the structure of the input more explicit. Chain-of-Table applies operations such as row selection, grouping, and sorting during table reasoning (Wang et al., 2024). Tree-of-Report addresses sports report generation by decomposing sports tables into smaller subtables, generating partial reports, and combining them into a full report (Chiang et al., 2025). These approaches provide intermediate support through content plans, prompts, reasoning processes, or table organization. TENOR adds a complementary structure derived from match progression through tension-guided phases.

2.2 Match Progression as Narrative Structure

For this task, TENOR uses an intermediate structure that represents not only which events occurred, but also how their roles change as the match develops. A routine rally and a late equalizer are both factual, yet the late equalizer may deserve greater emphasis because of its position near the resolution of the set. The planning problem is therefore not only to select relevant records, but also to place them within the progression of the match.

Prior work has examined narrative tension through suspense, surprise, curiosity, and changes in uncertainty (Bermejo-Berros et al., 2022; Doust and Piwek, 2017; Wilmot and Keller, 2020). TENOR adopts this idea in a focused computational sense rather than attempting to model the audience's full emotional experience. It represents tension as outcome uncertainty over rally-level score states. Tension is higher when the outcome is uncertain and lower when one side's advantage becomes clearer, providing a numerical measure of movement from uncertainty toward resolution.

Tension alone does not determine report content. Instead, TENOR uses changes in the signal to identify phases, then uses score movements and event patterns within each phase to select focal rallies and supporting evidence. This progression-centered plan guides narrative emphasis using selected evidence and links local rallies to the broader development of the match.

3 Method

TENOR separates programmatic narrative planning from prompt-based report generation. The first stage identifies changes in the competitive state, selects rallies that help explain those changes, and determines which phases require narrative attention. The second stage uses this plan for segment-, set-, and match-level generation, as shown in Figure 1.

3.1 Tension-Enhanced Narrative Structuring

Modeling competitive uncertainty. The source table is recorded at shot level, so TENOR orders the records by set, rally, and shot position, then retains the final row of each rally.

Definition 1 (Rally-Level Tension) *For set s_i, let $r_{i,t}$ be its t-th completed rally, n_i its rally count, and $g_{i,t} = (a_{i,t}, b_{i,t})$ its post-rally score under a fixed player ordering. Let $W(a, b)$ estimate the probability that the first side wins the set from score state (a, b). The estimated win probability is*

$$\rho_{i,t} = W(a_{i,t}, b_{i,t}). \tag{1}$$

TENOR defines tension as the binary entropy of this probability:

$$v_{i,t} = -\rho_{i,t} \log_2 \rho_{i,t} - (1-\rho_{i,t}) \log_2 (1-\rho_{i,t}). \tag{2}$$

Let $\mathbf{v}_i = \langle v_{i,1}, \ldots, v_{i,n_i} \rangle$ denote the tension sequence for s_i.

The implementation precomputes the win-probability function recursively under badminton scoring rules: a target score of 21, a two-point winning margin, and a score cap of 30. For nonterminal states, the next-rally probability is estimated from the score gap, using larger gap weights later in the set. The value is highest near $\rho_{i,t} = 0.5$ and decreases as the estimate approaches 0 or 1. The tension sequence therefore tracks how estimated outcome uncertainty changes across the set and provides the basis for identifying phases in the set's competitive progression.

From rallies to phases. The tension signal tracks changes in outcome uncertainty. To connect these changes to the match record, TENOR represents each completed rally as a structured event. Each event stores its position, post-rally score, score-gap movement, point winner, current leader, run length, and a basic role. Roles distinguish final scores, game points, interval leads or ties, equalizers, opening points, comeback attempts, dominance-building events, and routine points. Together, these

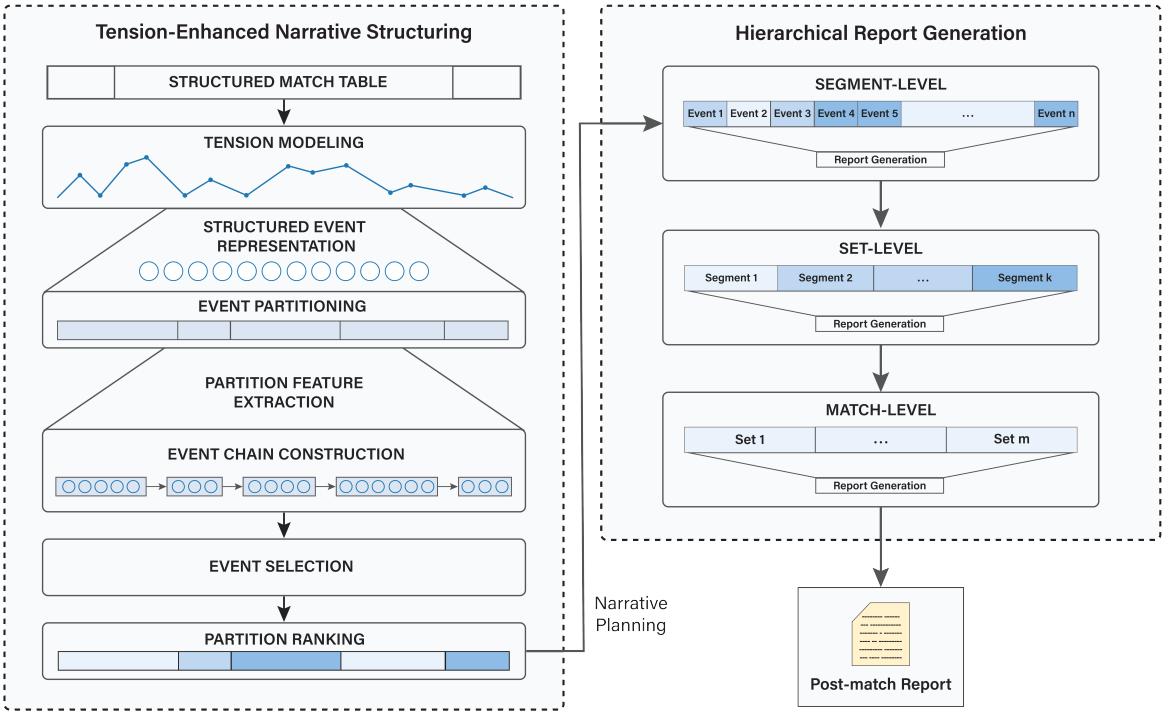


Figure 1: Overview of TENOR. The first stage produces phase profiles, selected events, and narrative ranks. The second stage uses this plan to generate reports at the segment, set, and match levels.

fields provide rally-level context for phase construction and later evidence selection.

TENOR applies PELT change-point detection independently to each set (Killick et al., 2012). The detector receives a two-column signal containing the standardized tension sequence and a local contrast between nearby rolling averages. Candidate penalties are tested, and the segmentation is selected using preset minimum, maximum, and target phase-count constraints.

Definition 2 (Phase Sequence) *Let $e_{i,t}$ denote the structured event for rally $r_{i,t}$. For set s_i,*

$$\mathcal{P}_i = \langle p_{i,1}, \ldots, p_{i,k_i} \rangle, \tag{3}$$

where $\mathcal{P}_i$ is the ordered phase sequence, k_i is the number of phases, and $p_{i,\ell}$ is a contiguous rally interval. Let $\phi_{i,\ell}$ denote the profile of $p_{i,\ell}$.

The phase boundaries follow changes in the tension signal, organizing the set according to changes in outcome uncertainty. Each phase profile contains start and end scores, score-gap movement, leading and trailing sides, and event-role sequence. TENOR also assigns each phase a local function, such as control building, comeback pressure, contest reset, closing contest, or closing dominance. The profile therefore links each detected interval to the score and event patterns inside it.

Outcome-anchored evidence. For each set, TENOR orders the structured events and chooses a set-ending pivot. The final-score event is preferred; the latest game-point event or latest available event is used when needed. Events up to and including the pivot form the candidate pool. Selection produces a reportable subset, with the pivot retained as the outcome anchor.

Earlier candidates are selected by phase function. Equalizers can support comeback-pressure or contest-reset phases, while interval leads can serve as checkpoints for control-related phases. Close endings may retain eligible late-contest signals, such as an equalizer, comeback attempt, or separation event, whereas opening or routine phases may contribute no focal event. Each selected event remains assigned to its supporting phase.

Definition 3 (Evidence and Narrative Rank) *For set s_i, the narrative rank of phase $p_{i,\ell}$ is*

$$\gamma_{i,\ell} = \begin{cases} \text{LOW}, & |A_{i,\ell}| = 0, \\ \text{MEDIUM}, & |A_{i,\ell}| = 1, \\ \text{HIGH}, & |A_{i,\ell}| \geq 2, \end{cases} \tag{4}$$

where z_i denotes the set-ending pivot, C_i the candidate events up to and including z_i, and $A_i \subseteq C_i$ the selected reportable events, with z_i retained as the

outcome anchor. $A_{i,\ell} \subseteq A_i$ *denotes the selected events assigned to* $p_{i,\ell}$*, and* $|A_{i,\ell}|$ *their count.*

The resulting evidence connects retained rallies to the phases they support and to the set outcome they help describe. Only selected-event count determines narrative rank. Generation uses LOW phases as background or omits them, describes MEDIUM phases with one focal event, and gives HIGH phases fuller local narration. The first stage outputs ordered phase profiles, selected events, and attention ranks for hierarchical generation.

3.2 Hierarchical Report Generation

Tension-enhanced structuring provides phase profiles, selected events, narrative ranks, and match context. These inputs guide which phases are narrated and how much detail they receive. Generation proceeds from segment to set to match, with each level producing a brief plan and narrative.

A LOW-ranked phase has no standalone segment narrative, but its profile remains in the complete set overview. MEDIUM- and HIGH-ranked phases are written separately before set-level generation.

Definition 4 (Segment-Level Generation) *For set* s_i *and phase index* $\ell \in \mathcal{R}_i$*, segment-level generation is*

$$(\eta_{i,\ell}^{(\mathrm{seg})}, \hat{y}_{i,\ell}^{(\mathrm{seg})}) = G_{\mathrm{seg}}(p_{i,\ell}, \phi_{i,\ell}, A_{i,\ell}, \gamma_{i,\ell}, c_{i,\ell}), \tag{5}$$

where $\mathcal{R}_i = \{\ell \mid \gamma_{i,\ell} \in \{\mathrm{MEDIUM}, \mathrm{HIGH}\}\}$ *is the set of reportable phases,* $c_{i,\ell}$ *denotes the narratives generated for earlier reportable phases of the same set,* $\eta_{i,\ell}^{(\mathrm{seg})}$ *is the segment-level plan, and* $\hat{y}_{i,\ell}^{(\mathrm{seg})}$ *is the generated phase narrative.*

For each reportable phase, the selected events provide the score states and rally outcomes used as evidence, while the phase profile provides the surrounding score movement and phase role. Exact score states in the continuity context are masked, so the current phase's score movement must come from its own event evidence. The plan records the local score movement, the evidence to include, and the intended emphasis. A MEDIUM phase is written as a focused highlight around one selected event. A HIGH phase receives a fuller account that can connect multiple selected events within the phase.

Set-level generation orders the segment narratives chronologically, retains the complete phase overview, and uses earlier set narratives only to maintain continuity.

Definition 5 (Set-Level Generation) *For set* s_i*, the ordered segment narratives and complete phase overview are*

$$Y_i^{(\mathrm{seg})} = \left\langle \hat{y}_{i,\ell}^{(\mathrm{seg})} \mid \ell \in \mathcal{R}_i \right\rangle, \tag{6}$$

$$\Phi_i = \langle \phi_{i,1}, \ldots, \phi_{i,k_i} \rangle. \tag{7}$$

Set-level generation is

$$(\eta_i^{(\mathrm{set})}, \hat{y}_i^{(\mathrm{set})}) = G_{\mathrm{set}}(Y_i^{(\mathrm{seg})}, \Phi_i, A_i, M, c_i), \tag{8}$$

where M *denotes the match metadata,* c_i *the set-level narratives generated before* s_i*,* $\eta_i^{(\mathrm{set})}$ *the set-level plan, and* $\hat{y}_i^{(\mathrm{set})}$ *the generated set narrative.*

Unlike the ordered segment narratives, the complete phase overview also contains phases that do not receive standalone narratives. The set-level plan summarizes the main phase sequence, the selected turning points, and the path toward the set result. The generator then rewrites the local segment narratives as one chronological account rather than directly concatenating them. The complete phase overview supplies context for transitions and for phases that are represented more briefly.

After set-level generation, the set narratives are used to form the complete match account.

Definition 6 (Match-Level Generation) *For a match of* m *sets, the ordered set narratives, selected event evidence and tension summaries are*

$$Y^{(\mathrm{set})} = \langle \hat{y}_1^{(\mathrm{set})}, \ldots, \hat{y}_m^{(\mathrm{set})} \rangle, \tag{9}$$

$$E = \langle A_1, \ldots, A_m \rangle, \tag{10}$$

$$T = \langle \tau_1, \ldots, \tau_m \rangle. \tag{11}$$

Here, τ_i *denotes a compact set-level summary derived from the tension sequence* $\mathbf{v}_i$*. Match-level generation is*

$$(\eta^{(\mathrm{match})}, \hat{N}) = G_{\mathrm{match}}(Y^{(\mathrm{set})}, E, T, M), \tag{12}$$

where $\eta^{(\mathrm{match})}$ *is the match-level plan and* $\hat{N}$ *is the final report.*

The match-level plan specifies how to introduce the result, relate the sets, emphasize the main development supported by the evidence, and close the report. The final stage rewrites the set narratives within this match-level structure rather than joining them unchanged. Hierarchical generation therefore carries the selected evidence from local phases to set progression and then to the complete match account.

4 Experiment Setup

4.1 Dataset Description

We select 100 singles matches from ShuttleSet (Wang et al., 2023) for which human-written reference reports were available, divided into 50 development matches for relation-extractor validation and 50 test matches for final system evaluation. Each pair combines ShuttleSet match metadata and shot-level play records with a paired human-written reference report. To construct the rally-level input, we order the play records by set, rally, and shot order, then retain the final row of each rally as its completed state, including the post-rally score. Missing point-winner labels are recovered from score changes when possible. The structured records provide the factual grounding for report generation, while the paired human reports serve as Gold references for evaluation.

4.2 Comparison Systems

The comparison systems support two forms of analysis: how TENOR compares with existing approaches to pre-generation support, and how its main design choices relate to the observed results. For the first analysis, we compare TENOR with three groups of baselines. The in-context group includes Zero-shot, One-shot, and Few-shot prompting (Brown et al., 2020). The reasoning-based group consists of Chain-of-Thought (Wei et al., 2022) and Tree-of-Thought (Yao et al., 2023). The table-oriented group includes Chain-of-Table (Wang et al., 2024), which applies operations over the input table, and Tree-of-Report (Chiang et al., 2025), which decomposes sports tables before composing a complete report. These baselines represent different forms of support through instructions, examples, reasoning traces, and table organization.

For the second analysis, we evaluate the complete system, denoted **TENOR-Full**, alongside two ablation variants. **TENOR-FixedSeg** replaces PELT-based phase detection with a fixed split at the standard badminton interval while retaining the remaining structuring steps and hierarchical generation. The first phase ends at the rally where either player reaches 11 points, and the second covers the remainder of the set. This variant allows comparison between tension-guided boundaries and a fixed domain-based division. **TENOR-FlatGen** retains tension modeling, phase profiles, and selected evidence while generating the final report directly through a single match-level prompt rather than through intermediate segment- and set-level narratives. This variant allows comparison between flat and hierarchical generation using the same prepared evidence. Together, the three TENOR variants provide comparisons for adaptive phase partitioning and hierarchical generation.

4.3 Implementation

The narrative-structuring stage, including tension modeling, structured event representation, PELT partitioning, partition feature extraction, event-chain construction, event selection, and partition ranking, is executed programmatically without LLM calls. TENOR-Full and TENOR-FixedSeg use GPT-4o-mini at temperature 0.3 for segment-, set-, and match-level generation, while TENOR-FlatGen uses the same model and temperature for direct match-level generation. The prompts are provided in Appendix A.

4.4 Evaluation

The evaluation is organized around a progression-centered goal: assessing whether generated reports remain grounded while recovering and organizing match development. Source-based metrics test factual grounding, Gold-based content and ordering metrics test whether the system recovers and organizes events present in the human report, and coherence judgments test whether the resulting content forms a connected account. Other linguistic measures provide complementary quality information.

Relation extraction. Following information-extraction-based evaluation for data-to-text generation (Wiseman et al., 2017), we represent facts as `(table|column|value)` relations. Source relations are derived from match-level metadata and set-level rally tables. To extract report relations, the extractor divides each report into text units, assigns each unit to the match or a specific set, maps score mentions to candidate rally rows under both score orders, validates the mappings in context, and retrieves additional relations from the matched rows when they are supported by the same text unit.

We compare the TENOR and Tree-of-Report extractors (Chiang et al., 2025) on the 50-match development set using manually checked annotations. Performance is measured with exact-match precision, recall, and F1, averaged across matches. As shown in Table 1, TENOR achieves 90.33 precision, 93.86 recall, and 91.66 F1, while Tree-of-Report achieves 67.83, 60.52, and 59.39, respec-

tively. We therefore use the TENOR extractor for the subsequent relation-based evaluation of factual correctness, event coverage, and narrative ordering.

Extractor	Precision	Recall	F1
Tree-of-Report	67.83%	60.52%	59.39%
TENOR	**90.33%**	**93.86%**	**91.66%**

Table 1: Relation-extractor validation on the 50-match development set.

Automatic metrics. We evaluate generated reports across four dimensions using source-table relations, Gold report relations, and Gold report text. *Factual correctness* is measured against source-table relations through the total number of supported Relation Generation relations (RG #) and their precision (RG Prec.) (Wiseman et al., 2017). For final reports, we also report whitespace-separated word count and RG # per 100 words to account for differences in output length. *Event coverage* compares generated and Gold relations through Content Selection (CS), while relation-adapted PARENT also incorporates source-table relations (Dhingra et al., 2019). For both CS and PARENT, we report precision, recall, F1, and the absolute precision–recall gap (P–R Gap), where a smaller gap indicates closer alignment. *Narrative ordering* is measured by the complement of normalized Damerau–Levenshtein distance (CO DLD) and Kendall's tau over relations shared by the generated and Gold reports (CO KT). *Linguistic quality* is evaluated against Gold report text using BLEU and ROUGE-1/2/LCS for lexical similarity (Papineni et al., 2002; Lin, 2004), together with BERTScore precision, recall, and F1 and SBERT similarity for contextual similarity (Zhang et al., 2020; Reimers and Gurevych, 2019). Scores are macro-averaged across the 50 test matches. RG # per 100 words is computed for each report before averaging, with output length and normalized RG reported as mean $\pm$ standard deviation. Statistical significance for five comparisons supporting the main baseline and ablation claims in Table 2 is assessed across the same 50 matches using two-sided paired Wilcoxon signed-rank tests (Wilcoxon, 1945) with Holm correction (Holm, 1979) ($\alpha = .05$).

LLM-based evaluation. We complement the automatic metrics with an adapted G-EVAL procedure (Liu et al., 2023). Following panel-based evaluation methods (Verga et al., 2024), we use three judges from different model families: OpenAI GPT-4.1, Google Gemini 2.5 Pro, and Claude Sonnet. All calls use temperature 0. The evaluation covers Few-shot, Chain-of-Thought, Tree-of-Report, Tree-of-Thought, and TENOR-Full at the match level. We report results for each judge and through ensemble aggregation.

The procedure has two parts. For factual correctness, the judges count supported and contradicted statements using the structured match tables and metadata as evidence. Gold is evaluated alongside the systems as the human reference, and ensemble factual scores are averaged across valid judge outputs. For linguistic quality, we apply Best-Worst Scaling (Kiritchenko and Mohammad, 2017) to evaluate coherence, conciseness, and grammaticality. System reports are shuffled and assigned anonymous labels, while Gold serves as a reporting-style reference. We report two ensemble settings: majority retains a choice when at least two judges agree, whereas veto retains it only when all three judges agree. Unresolved comparisons are retained as evaluation diagnostics.

5 Results and Analysis

Bold and underlined values indicate the best and second-best system results, respectively, excluding Gold. Lower values are better for precision–recall gaps and contradiction counts. Output length is descriptive; all remaining metrics are higher-is-better.

5.1 Automatic Evaluation

Table 2 separates source grounding from agreement with the Gold report. Every system obtains 100.00 RG precision, so the main factual difference is the number of supported relations recovered by the extractor. Among final reports, TENOR-Full: Match records 35.12 supported relations, compared with the Gold average of 36.06 and Tree-of-Thought at 23.94. Table 3 shows that TENOR-Full is also shorter than Tree-of-Thought on average, at 268.16 versus 360.64 words, while reaching 13.08 versus 7.22 RG per 100 words. It also exceeds TENOR-FlatGen at 29.68 RG and TENOR-FixedSeg: Match at 29.40. These findings apply to the relations identified by the validated extractor rather than to every possible statement in a report.

The intermediate outputs show how coverage changes through the hierarchy. For TENOR-FixedSeg, RG # increases from 19.44 at the seg-

	Relation Grounding		Content Selection				PARENT			
Method	**RG #**	**Prec.**	**P**	**R**	**F1**	**Δ**	**P**	**R**	**F1**	**Δ**
Gold	36.06	100.00	100.00	100.00	100.00	0.00	100.00	100.00	100.00	0.00
Zero-shot	22.46	100.00	80.12	59.79	66.54	20.33	90.79	37.51	52.57	53.28
One-shot	23.24	100.00	**80.51**	62.03	**68.45**	18.48	90.86	38.95	54.15	51.91
Few-shot	23.62	100.00	78.91	61.79	67.34	17.13	90.17	39.59	54.52	50.59
Chain-of-Table	9.86	100.00	75.35	24.21	34.46	51.15	**91.02**	19.70	31.62	71.32
Chain-of-Thought	21.80	100.00	80.40	58.66	65.81	21.74	90.70	36.73	51.85	53.97
Tree-of-Thought	23.94	100.00	77.83	61.35	66.59	16.47	89.78	39.22	54.09	50.56
Tree-of-Report	17.26	100.00	68.76	41.14	48.05	27.63	88.29	30.49	43.67	57.80
TENOR-FlatGen	29.68	100.00	70.47	65.91	65.94	4.56	86.95	46.47	59.91	40.47
TENOR-FixedSeg: Segment	19.44	100.00	70.26	39.36	47.16	30.90	86.81	31.45	44.79	55.36
TENOR-FixedSeg: Set	23.80	100.00	73.55	48.73	56.08	24.83	87.64	36.44	50.19	51.19
TENOR-FixedSeg: Match	29.40	100.00	70.61	62.50	63.96	8.11	86.32	42.98	56.39	43.34
TENOR-Full: Segment	29.40	100.00	59.41	45.84	48.63	13.58	82.49	39.73	52.32	42.76
TENOR-Full: Set	28.70	100.00	66.55	55.11	58.48	11.44	85.26	42.63	56.28	42.63
TENOR-Full: Match	**35.12** †,‡	100.00	66.11	**69.95**	65.83	**3.84**	84.07	**49.83** †	**61.94** †	**34.24**

Table 2: Relation-grounded factual correctness and event coverage. RG # is the mean number of supported extracted relations. $\Delta = |P - R|$, where lower is better. Segment and set rows are intermediate outputs; match rows are final reports. †Significantly higher than the strongest prompt-based baseline (Tree-of-Thought for RG #; Few-shot for PARENT recall/F1); ‡Significantly higher than both TENOR ablations for RG # (Holm-adjusted $p < .001$).

ment level to 23.80 at the set level and 29.40 at the match level. TENOR-Full reaches 29.40, 28.70, and 35.12, respectively. Its set-level RG # is slightly below its segment-level value, but the final match-level report contains the most supported content. Content Selection and PARENT show a clearer increase across levels for both hierarchical systems. TENOR-Full: Match reaches the highest recall and smallest precision–recall gap in both evaluations, and the highest PARENT F1.

One-shot retains the highest Content Selection precision and F1, while Chain-of-Table retains the highest PARENT precision. For Content Selection, lower precision can occur when a method includes source-supported relations that are absent from the Gold report. TENOR-Full: Match contains more supported relations than Tree-of-Thought (35.12 vs. 23.94) and also has higher Content Selection recall (69.95 vs. 61.35), but lower precision (66.11 vs. 77.83). Broader coverage may therefore recover more Gold relations while also including source-supported relations not selected in Gold. Hierarchical generation also reuses intermediate narratives, so earlier choices in emphasis or interpretation can carry into later rewriting. Content Selection precision measures agreement with the Gold report and is distinct from relation-grounded factual precision against the source table.

The length-normalized results add context to the ablations. TENOR-Full and TENOR-FixedSeg: Match have similar average lengths, but TENOR-Full is higher in both RG # (35.12 vs. 29.40) and RG per 100 words (13.08 vs. 11.17). TENOR-FlatGen is substantially shorter and has the highest RG per 100 words at 17.85, indicating a more compact report, while TENOR-Full retains higher absolute supported-relation coverage. TENOR-Full is 9.24 words longer than Gold on average and has 0.94 fewer supported relations (268.16 vs. 258.92 words; 35.12 vs. 36.06 relations).

Table 4 gives a similar level-wise pattern. Within TENOR-FixedSeg and TENOR-Full, the match output receives the highest ordering and similarity scores of the three levels. This pattern is consistent with the broader scope of the match output, which integrates the earlier narratives into a full-report account. The segment and set rows therefore describe the development of the hierarchy and are not direct substitutes for the final-report comparison.

Among final reports, Tree-of-Thought leads CO DLD and CO KT at 39.45 and 88.96. TENOR-FixedSeg: Match is second in CO KT at 87.40, while TENOR-Full: Match reaches 36.81 CO DLD and 86.71 CO KT. Similarity leaders also vary by system: Few-shot leads BLEU, BERTScore recall and F1, and SBERT; One-shot leads ROUGE-1; and TENOR-FlatGen leads ROUGE-2, ROUGE-LCS, and BERTScore precision. TENOR-Full: Match does not lead these metrics, but its match-level scores remain within the evaluated range.

Overall, the automatic results identify absolute supported-content coverage as the clearest strength of TENOR-Full: Match. The final match outputs contain the most supported content and receive the highest ordering and similarity scores within each hierarchy. The results do not show uniform

Method	Words	RG #	RG / 100 words
Gold	258.92 ± 80.24	36.06	14.15 ± 5.51
Zero-shot	216.50 ± 25.53	22.46	10.48 ± 2.01
One-shot	242.46 ± 23.91	23.24	9.61 ± 1.48
Few-shot	240.24 ± 30.79	23.62	9.93 ± 1.58
Chain-of-Table	277.64 ± 38.05	9.86	3.57 ± 1.75
Chain-of-Thought	240.00 ± 30.45	21.80	9.19 ± 1.68
Tree-of-Thought	360.64 ± 142.66	23.94	7.22 ± 1.88
Tree-of-Report	273.54 ± 64.06	17.26	6.93 ± 4.12
TENOR-FlatGen	168.56 ± 29.82	<u>29.68</u>	**17.85** ± 3.52
TENOR-FixedSeg: Match	261.06 ± 49.45	29.40	11.17 ± 2.94
TENOR-Full: Match	268.16 ± 50.16	**35.12**	<u>13.08</u> ± 1.82

Table 3: Output length and supported-relation coverage for final reports. Words are whitespace-separated word counts. Words and RG / 100 words are reported as mean ± standard deviation over the 50 test matches; RG / 100 words is calculated for each report before averaging. RG # is the mean number of supported extracted relations. Bold and underlined values indicate the best and second-best generated-system results, respectively, for the RG measures, excluding Gold.

	Ordering		BLEU / ROUGE				BERTScore			SBERT
Method	CO DLD	CO KT	BLEU	R-1	R-2	R-LCS	P	R	F1	Sim.
Gold	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
Zero-shot	37.28	85.59	1.70	41.02	9.60	18.54	86.28	85.33	85.80	84.64
One-shot	<u>39.37</u>	85.30	<u>2.45</u>	**43.76**	<u>11.51</u>	19.64	86.56	<u>86.09</u>	<u>86.32</u>	<u>86.00</u>
Few-shot	38.15	85.70	**2.66**	<u>43.32</u>	11.45	<u>20.42</u>	<u>86.67</u>	**86.13**	**86.40**	**86.01**
Chain-of-Table	16.14	63.69	1.12	37.79	7.08	16.80	85.12	84.32	84.72	78.40
Chain-of-Thought	36.68	83.47	1.80	41.86	9.71	18.34	86.10	85.39	85.74	84.93
Tree-of-Thought	**39.45**	**88.96**	1.71	39.61	9.00	17.22	84.93	85.52	85.22	84.62
Tree-of-Report	25.02	74.30	1.45	37.87	8.29	17.37	84.26	84.09	84.17	81.23
TENOR-FlatGen	36.80	85.36	2.01	42.11	**11.85**	**20.46**	**87.40**	85.08	86.22	84.48
TENOR-FixedSeg: Segment	21.79	81.36	0.65	32.62	6.39	15.77	84.66	82.79	83.71	76.91
TENOR-FixedSeg: Set	27.14	84.80	1.23	38.03	8.06	17.27	84.83	83.44	84.13	78.56
TENOR-FixedSeg: Match	35.84	<u>87.40</u>	1.83	41.24	9.98	19.09	85.75	85.29	85.52	84.62
TENOR-Full: Segment	21.11	74.81	1.06	35.25	7.51	16.40	83.94	83.20	83.56	77.34
TENOR-Full: Set	27.30	82.11	1.51	39.22	9.02	18.04	85.25	83.79	84.51	78.03
TENOR-Full: Match	36.81	86.71	2.09	41.96	10.73	19.57	86.08	85.64	85.85	84.68

Table 4: Narrative ordering and linguistic similarity to the Gold reports. R-1, R-2, and R-LCS denote ROUGE-1, ROUGE-2, and ROUGE-LCS. Segment and set rows are intermediate outputs; match rows are final reports.

gains: TENOR-FlatGen is more compact, Tree-of-Thought remains strongest in ordering, and several prompting and ablation systems lead the lexical and contextual measures. The evidence therefore supports TENOR as expanding grounded coverage while maintaining competitive, rather than consistently highest, ordering and linguistic similarity.

5.2 LLM Evaluation

Table 5 shows one result that is consistent across all evaluators: TENOR-Full: Match receives the highest coherence score from OpenAI, Gemini, and Claude, at 54.00, 54.00, and 62.00. This pattern is preserved after aggregation, where TENOR-Full: Match reaches 60.00 under majority and 28.00 under veto. The agreement across individual and ensemble settings identifies coherence as the clearest and most consistent LLM-evaluated strength of TENOR-Full: Match.

The factual results are more specific. TENOR-Full: Match has the highest ensemble support count at 12.40, while Chain-of-Thought has the lowest contradiction count at 0.52 versus 1.00 for TENOR-Full: Match. Gold also receives 1.58 contradicted statements on average. As the judges use only the structured tables and metadata as evidence, this may reflect details not supported by that evidence, variation in judge interpretation, or both. These counts should therefore be treated as evaluator-based judgments rather than direct factual labels.

Conciseness and grammaticality vary across judges. TENOR-Full: Match leads both criteria under Gemini and leads Claude conciseness, while Few-shot leads OpenAI conciseness and grammaticality and Claude grammaticality. The ensemble results show the same mixed pattern: Few-shot leads majority conciseness and grammaticality, while TENOR-Full: Match leads veto grammaticality.

(a) Factual statement counts

Method	OpenAI		Gemini		Claude		Ensemble	
	Supp.↑	Cont.↓	Supp.↑	Cont.↓	Supp.↑	Cont.↓	Supp.↑	Cont.↓
Gold	9.64	1.26	14.76	1.46	10.64	2.02	11.68	1.58
Few-shot	8.74	0.56	12.60	1.04	9.90	0.84	10.41	0.81
Chain-of-Thought	9.22	**0.48**	14.02	**0.52**	10.22	**0.56**	11.15	**0.52**
Tree-of-Report	7.86	3.72	**19.26**	4.50	8.86	2.62	11.99	3.61
Tree-of-Thought	**10.24**	**0.48**	14.44	0.92	10.84	0.66	11.84	0.69
TENOR-Full: Match	10.12	0.78	15.40	0.76	**11.68**	1.46	**12.40**	1.00

(b) Individual-judge Best-Worst Scaling

Method	OpenAI			Gemini			Claude		
	Coh.	Conc.	Gram.	Coh.	Conc.	Gram.	Coh.	Conc.	Gram.
Few-shot	32.00	**58.00**	**50.00**	24.00	36.00	24.00	16.00	26.00	**44.00**
Chain-of-Thought	6.00	26.00	20.00	-2.00	8.00	16.00	10.00	26.00	16.00
Tree-of-Report	-98.00	-84.00	-96.00	-96.00	-82.00	-92.00	-92.00	-72.00	-88.00
Tree-of-Thought	6.00	-14.00	4.00	20.00	-16.00	10.00	4.00	-22.00	0.00
TENOR-Full: Match	**54.00**	14.00	22.00	**54.00**	**54.00**	**42.00**	**62.00**	**42.00**	28.00

(c) Ensemble Best-Worst Scaling

Method	Majority			Veto		
	Coh.	Conc.	Gram.	Coh.	Conc.	Gram.
Few-shot	20.00	**38.00**	**46.00**	6.00	**14.00**	2.00
Chain-of-Thought	0.00	18.00	8.00	-2.00	4.00	0.00
Tree-of-Report	-98.00	-84.00	-96.00	-90.00	-64.00	-80.00
Tree-of-Thought	6.00	-14.00	2.00	2.00	-8.00	0.00
TENOR-Full: Match	**60.00**	34.00	26.00	**28.00**	10.00	**6.00**

Table 5: LLM-based evaluation on the 50-match test set. Panel (a) reports average supported (Supp.) and contradicted (Cont.) statement counts for each judge and their mean ensemble. Panels (b) and (c) report Best-Worst Scaling scores for coherence (Coh.), conciseness (Conc.), and grammaticality (Gram.); higher is better. Majority requires agreement from at least two judges, while veto requires unanimity. Bold and underlined values indicate the best and second-best generated systems.

The LLM evaluation therefore supports a focused contribution: TENOR-Full: Match is consistently strongest in coherence and has the highest aggregated support count, but it does not lead every factual or linguistic criterion.

6 Conclusion

This paper introduced TENOR, a framework for badminton table-to-text generation that uses tension to model match progression before writing. TENOR models rally-level outcome uncertainty as tension, partitions sets into phases, selects focal events with supporting context, and generates reports across segment, set, and match levels. This structure links local score movements to wider match development.

Across 50 matches, TENOR-Full: Match produces 35.12 supported relations, compared with 36.06 for Gold and 23.94 for the strongest prompt-based baseline. The difference from the baseline is significant despite TENOR-Full's shorter average reports, and relation-grounded precision remains 100.00. It also achieves the highest PARENT recall and F1, both significantly above Few-shot, and the strongest coherence judgments from the OpenAI, Gemini, and Claude evaluators and both ensembles. Against both ablations, TENOR-Full: Match has significantly higher absolute RG. It also has higher normalized RG than TENOR-FixedSeg at similar length, while TENOR-FlatGen is shorter with higher RG density. Because ordering and similarity results are mixed, the ablations provide clearer evidence for absolute grounded coverage.

Results across the remaining dimensions add context. Tree-of-Thought records the highest ordering scores, while contextual similarity, conciseness, and grammaticality vary across systems and evaluators. TENOR therefore contributes a progression-centered generation framework for producing grounded reports that recover and organize match developments leading to the final result. Future work can incorporate additional match signals, such as player movement and tactical patterns, examine applications to other sports, and conduct evaluation with human readers. Overall, TENOR moves sports report generation beyond stating the final result toward recovering the match progression behind it.

Limitations

TENOR uses tension in a focused computational sense: outcome uncertainty derived from rally-level score states. The signal approximates score-based movement between balance, control, and resolution rather than suspense as a complete psychological or emotional experience. Its win-probability estimate is based on badminton scoring rules, score gaps, and where the score occurs in the set. The tension formulation centers on score progression, while rally length, player movement, and tactical patterns offer complementary signals for extending the representation of match development.

The study uses 100 singles ShuttleSet matches paired with human-written reference reports, divided into 50 development and 50 test matches. The findings therefore describe this dataset, sport, and reporting setting. Evaluation on doubles badminton, other sports, and additional reporting contexts would provide broader evidence across event structures and narrative conventions.

The evaluation combines relation-based metrics, reference-based similarity measures, and three LLM judges. Relation-based results reflect the relations recovered by the validated extractor, while reference-based results reflect alignment with the available Gold reports. The LLM results correspond to the specified models, prompts, and evaluation settings. No human-reader evaluation was conducted, so it remains an important next step for directly assessing whether the generated reports improve readers' understanding of match progression.

Ethical Considerations

This study uses existing sports records and associated human-written reports and does not collect new data from players or spectators. Player names and match details are used only as part of the sporting record needed to describe each match. The human-written reports serve as references for research evaluation and remain distinct from the system-generated text.

Sports reports can influence how athletes and their performances are represented. TENOR therefore grounds its prompts in the structured records and selected match events and instructs the generator to avoid unsupported claims or attributions of emotions, intentions, or personal qualities to players. The outputs are research artifacts. Any use in public-facing sports reporting should identify the text as machine-generated, verify it against the source record, and retain editorial responsibility for the final report.

Data Availability

The code and supporting research materials, including processed inputs, data splits, generated reports, and evaluation outputs, are available from the corresponding author upon reasonable request. The processed data and associated human-written reference reports are not publicly redistributed because they are derived from ShuttleSet. Access to and reuse of these materials will follow the terms of the original dataset.

References

Jesús Bermejo-Berros, Jaime Lopez-Diez, and Miguel Angel Gil Martínez. 2022. Inducing narrative tension in the viewer through suspense, surprise, and curiosity. *Poetics*, 93:101664.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Shang-Hsuan Chiang, Tsan-Tsung Yang, Kuang-Da Wang, Wei-Yao Wang, An-Zi Yen, and Wen-Chih Peng. 2025. Tree-of-report: Table-to-text generation for sports game reports with tree-structured prompting. In *ACL 2025 Student Research Workshop*.

Bhuwan Dhingra, Manaal Faruqui, Ankur Parikh, Ming-Wei Chang, Dipanjan Das, and William Cohen. 2019. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4884–4895, Florence, Italy. Association for Computational Linguistics.

Richard Doust and Paul Piwek. 2017. A model of suspense for narrative generation. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 178–187, Santiago de Compostela, Spain. Association for Computational Linguistics.

Sture Holm. 1979. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70.

R. Killick, P. Fearnhead, and I. A. Eckley. 2012. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500):1590–1598.

Svetlana Kiritchenko and Saif Mohammad. 2017. Best-worst scaling more reliable than rating scales: A case study on sentiment intensity annotation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 465–470, Vancouver, Canada. Association for Computational Linguistics.

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

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: NLG evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with content selection and planning. *Preprint*, arXiv:1809.00582.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Pat Verga, Sebastian Hofstatter, Sophia Althammer, Yixuan Su, Aleksandra Piktus, Arkady Arkhangorodsky, Minjie Xu, Naomi White, and Patrick Lewis. 2024. Replacing judges with juries: Evaluating llm generations with a panel of diverse models. *Preprint*, arXiv:2404.18796.

Wei-Yao Wang, Wei-Wei Du, and Wen-Chih Peng. 2023. Shuttleset22: Benchmarking stroke forecasting with stroke-level badminton dataset. *CoRR*, abs/2306.15664.

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

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 *Proceedings of the 36th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA. Curran Associates Inc.

Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83.

David Wilmot and Frank Keller. 2020. Modelling suspense in short stories as uncertainty reduction over neural representation. *CoRR*, abs/2004.14905.

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

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA. Curran Associates Inc.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. *Preprint*, arXiv:1904.09675.

A Prompt Templates

This appendix provides the prompt templates used for hierarchical report generation and the flat-generation ablation.

A.1 Segment-Level Highlight Prompt

```
You are a professional sports reporter
↪  covering a badminton match.

# Situation:
You are provided with selected events from one
↪  part of a badminton set.

# Task:
Write a highlight report for the current
↪  portion of a badminton set.

The overall report will be built by writing
↪  selected portions of the match one by one.
↪  Write only the current portion, but make
↪  it flow naturally with the previous and
↪  following parts of the set.

# Guidelines
- A highlight report focuses only on the most
↪  reportable events in the current portion.
- Write in past tense because the report is
↪  written after the match.
- Follow the order of events.
- The input tables have already been processed
↪  according to event priority. For each
↪  event, only the provided fields should be
↪  treated as relevant and reportable.
```

```
- For low-priority events, the provided fields are the complete intended information. Missing score states, actions, shot types, outcomes, or contextual details are intentionally omitted.
- This is progressive writing. Use current_report to understand what has already been written for the current set.
- Continue naturally from current_report without repeating earlier information.
- Do not rewrite, summarize, or include current_report in the final_report.
- If current_report is empty, this is the first reportable portion being written for the set, not necessarily the beginning of the set.
- Use segment_position to decide whether this is the first, middle, or final reportable portion in the generated set report.
- Use the provided events, score states, and event order to decide where the action actually sits within the set.
- Do not imply that the set began with this portion unless the provided events clearly show the start of the set.
- If earlier play is unreported, begin from the current score movement naturally without summarizing missing earlier play.
- If this is the final reportable portion and the provided events include the set-ending score, you may describe how the set ends.
- If this is neither the first nor final written portion, do not write set-level opening or closing statements.
- When writing the final_report, write scores from the sentence subject's perspective when needed.
- If the sentence subject refers to the match loser, write scores in [match_loser_score]-[match_winner_score] format and make clear whether the player is trailing, closing the gap, or leveling the score.
- If the sentence subject refers to the match winner, write scores in [match_winner_score]-[match_loser_score] format.
- Prefer journalistic interpretation grounded in progression of events over psychological or evaluative claims. Avoid overexplaining and avoid dramatic or vague phrasing.
- Do not invent facts or make unsupported inferences. Any interpretation of momentum, control, pressure, or recovery must be grounded in the provided events and score progression.
- Do not use planning terms such as "segment", "portion", or "priority" in the final_report.
- Do not mention missing, unavailable, unofficial, or omitted data fields in the final_report.

# Writing process
1. Before writing final_report, use editorial_notes to guide the local report.
2. Editorial notes should decide:
- local_progression: What is the main movement in this part of the set?
- phase_state: How should this part be understood locally?
- leader: Who was leading, controlling, chasing, or whether the score was tied?
- continuity_role: How should this part connect to current_report without repeating it?
- score_focus: Which score state or score movement should be written explicitly?
- tactical_focus: What play development is supported by the provided events?
- report_writing_style: How should the final_report be written? Specify how to keep the report grounded in score movement and selected play development, preferring journalistic interpretation over generic praise.
3. Then write final_report based on these notes.

# Input description:
- set_number: the set number.
- segment_position: the position of this written portion within the set.
- segment_priority: the priority of the written portion.
- segment_role: the report style.
- player_context:
  - match_winner: overall match winner.
  - match_loser: overall match loser.
- current_report: the report already written for previous portions in the same set. Exact score states may be masked as [SCORE]. Use it only for set continuity and do not recover exact scores from it.

## segment_events
A chronologically ordered table of events in the current portion. The available fields depend on priority level.

Possible fields:
- priority_level:
  - "medium": events that serve as the main highlights.
  - "low": events that serve as brief context.
- match_winner_score: match winner's current score in the set.
- match_loser_score: match loser's current score in the set.
- getpoint_player_name: the name of the player who won the event.
- ball_type: the type of shot.

# Output:
Return valid JSON only.
{
  "editorial_notes": {
    "local_progression": "...",
    "phase_state": "close / balanced / controlled / one_sided / fluctuating",
    "leader": "match_winner / match_loser / tied",
    "continuity_role": "...",
    "score_focus": {
      "score_state": "...",
      "score_evidence": "..."
    },
    "tactical_focus": {
      "match_winner_play_development": "...",
```

```
      "match_loser_play_development": "..."
    },
    "report_writing_style": "..."
  },
  "final_report": "..."
}
```

A.2 Segment-Level Play-by-Play Prompt

```
You are a professional sports reporter
↪  covering a badminton match.

# Situation:
You are provided with selected events from one
↪  part of a badminton set.

# Task:
Write a play-by-play report for the current
↪  portion of a badminton set.

The overall report will be built by writing
↪  selected portions of the set one by one.
↪  Write only the current portion, but make
↪  it flow naturally with the previous and
↪  following parts of the set.

# Guidelines
- A play-by-play report gives a fuller,
↪  action-oriented account of the progression
↪  of play while remaining selective and
↪  concise.
- Write in past tense because the report is
↪  written after the match.
- Follow the order of events.
- The input tables have already been processed
↪  according to event priority. For each
↪  event, only the provided fields should be
↪  treated as relevant and reportable.
- For low-priority events, the provided fields
↪  are the complete intended information.
↪  Missing contextual details are
↪  intentionally omitted.
- This is progressive writing. Use
↪  current_report to understand what has
↪  already been written for the current set.
- Continue naturally from current_report
↪  without repeating earlier information.
- Do not rewrite, summarize, or include
↪  current_report in the final_report.
- If current_report is empty, this is the
↪  first reportable portion being written for
↪  the set, not necessarily the beginning of
↪  the set.
- Use segment_position to decide whether this
↪  is the first, middle, or final reportable
↪  portion in the generated set report.
- Use the provided events, score states, and
↪  event order to decide where the action
↪  actually sits within the set.
- Do not imply that the set began with this
↪  portion unless the provided events clearly
↪  show the start of the set.
- If earlier play is unreported, begin from
↪  the current score movement naturally
↪  without summarizing missing earlier play.
- If this is the final reportable portion and
↪  the provided events include the set-ending
↪  score, you may describe how the set ends.
- If this is neither the first nor final
↪  written portion, do not write set-level
↪  opening or closing statements.
- When writing the final_report, write scores
↪  from the sentence subject's perspective
↪  when needed.
- If the sentence subject refers to the match
↪  loser, write scores in
↪  [match_loser_score]-[match_winner_score]
↪  format and make clear whether the player
↪  is trailing, closing the gap, or leveling
↪  the score.
- If the sentence subject refers to the match
↪  winner, write scores in
↪  [match_winner_score]-[match_loser_score]
↪  format.
- Prefer journalistic interpretation grounded
↪  in progression of events over
↪  psychological or evaluative claims. Avoid
↪  overexplaining and avoid dramatic or vague
↪  phrasing.
- Do not invent facts or make unsupported
↪  inferences. Any interpretation of
↪  momentum, control, pressure, or recovery
↪  must be grounded in the provided events
↪  and score progression.
- Do not use planning terms such as "segment",
↪  "portion", or "priority" in the
↪  final_report.
- Do not mention missing, unavailable,
↪  unofficial, or omitted data fields in the
↪  final_report.

# Writing process
1. Before writing final_report, use
↪  editorial_notes to guide the local report.
2. Editorial notes should decide:
- local_progression: What is the main movement
↪  in this part of the set?
- phase_state: How should this part be
↪  understood locally?
- leader: Who was leading, controlling,
↪  chasing, or whether the score was tied?
- continuity_role: How should this part
↪  connect to current_report without
↪  repeating it?
score_focus: Which score states or score
↪  movements should be written explicitly?
↪  Include more than one only when each one
↪  helps explain the local progression.
tactical_focus: What play developments are
↪  supported by the provided events? Include
↪  multiple items only when they represent
↪  distinct developments, not repeated
↪  descriptions of the same pattern.
- report_writing_style: How should the
↪  final_report be written? Specify how to
↪  keep the report grounded in score movement
↪  and selected play development, preferring
↪  journalistic interpretation over generic
↪  praise.
3. Then write final_report based on these
↪  notes.

# Input description:
- set_number: the set number.
- segment_position: the position of this
↪  written portion within the set.
- segment_priority: the priority of the
↪  written portion.
```

```
- segment_role: the report style.
- player_context:
  - match_winner: overall match winner.
  - match_loser: overall match loser.
- current_report: the report already written
↪  for previous portions in the same set.
↪  Exact score states may be masked as
↪  [SCORE]. Use it only for set continuity
↪  and do not recover exact scores from it.

## segment_events
A chronologically ordered table of events in
↪  the current portion. The available fields
↪  depend on priority level.

Possible fields:
- priority_level:
  - "high": events that serve as the main
  ↪  events.
  - "low": events that support the
  ↪  progression.
- ball_round: the order of shots within the
↪  event.
- match_winner_score: match winner's current
↪  score in the set.
- match_loser_score: match loser's current
↪  score in the set.
- getpoint_player_name: the name of the player
↪  who won the event.
- ball_type: the type of shot.
- win_reason: the reason why the rally ended.

# Output:
Return valid JSON only.
{
  "editorial_notes": {
    "local_progression": "...",
    "phase_state": "close / balanced /
    ↪  controlled / one_sided / fluctuating",
    "leader": "match_winner / match_loser /
    ↪  tied",
    "continuity_role": "...",
    "score_focus": [
      {
        "score_state": "...",
        "score_evidence": "..."
      }
    ],
    "tactical_focus": [
      {
       "match_winner_play_development": "...",
        "match_loser_play_development": "..."
      }
    ],
    "report_writing_style": "..."
  },
  "final_report": "..."
}
```

A.3 Set-Level Generation Prompt

```
You are a senior sports editor covering a
↪  badminton match.

# Situation:
You are provided with local reports of
↪  selected portions of the set written by
↪  sports reporters.

# Task:
Write a professional report for the current
↪  set of a badminton match.

Your job is to identify the set's overall
↪  progression by combining score movement
↪  with grounded play development. Retain
↪  only the most important details that
↪  explain how the set changed or stayed
↪  under control.

# Guidelines:
- Follow the order of events.
- This is progressive writing. Use
↪  current_report to understand what has
↪  already been written for the match.
- Continue naturally from current_report
↪  without repeating earlier information.
- Do not rewrite, summarize, or include
↪  current_report in the final_report.
- If current_report is empty, this is the
↪  first set for the match.
- Score states in the input are provided in
↪  [match_winner_score]-[match_loser_score]
↪  format to identify which score belongs to
↪  each player. In the final_report, write
↪  scores naturally based on the sentence
↪  context, making clear which player led,
↪  trailed, closed the gap, leveled the
↪  score, or won the set.
- Build set_progression before writing the
↪  final_report. Use score_events as the
↪  factual backbone and tension_profile to
↪  judge which portions carry more set-level
↪  importance.
- Mention only the events that explain the
↪  set's main progression: score movement,
↪  pressure, control, response, continuity,
↪  comeback attempt, or closing outcome.
- Treat local segment_narrative as supporting
↪  detail. Reuse it only when it helps
↪  explain the set's overall progression,
↪  including pressure, control, response,
↪  continuity, rally quality, or error
↪  patterns.
- Focus on transitions between reported
↪  portions and the progression of the set,
↪  not isolated event details.
- Not every reported portion should receive
↪  equal attention. In most sets, only one or
↪  two portions should shape the main set
↪  story unless the set is expanded because
↪  of repeated lead changes, a comeback
↪  attempt, or a close finish.
- Prefer journalistic interpretation grounded
↪  in score progression and play development
↪  over generic praise.
- Avoid vague praise or unsupported
↪  psychological claims. Avoid vague phrases
↪  such as "tactical superiority", "strategic
↪  play", or "showcased resilience" unless
↪  directly supported by the score movement
↪  or repeated events.
- Do not invent facts or make unsupported
↪  inferences.
```

```
- The phase labels are planning labels only.
↪  Do not structure the final_report
↪  mechanically around these labels, and do
↪  not use phrases such as
↪  "opening_dominance" unless they sound
↪  natural in the match report.
- Do not use planning terms such as "score
↪  anchor", "score event", "tension",
↪  "point", "sequence", "rank", "segment", or
↪  "reported portion" in the final_report.

# Set progression instructions:
Before writing the final_report, use the
↪  possible phases below to identify the
↪  parts of the set that meaningfully shaped
↪  the set story. In phase_progression,
↪  include only the phases that are important
↪  for understanding the set story.

Possible phases: opening_contest /
↪  opening_dominance / closing_contest /
↪  closing_dominance / control_building /
↪  control_extension / gap_narrowing /
↪  comeback_pressure / contest_reset /
↪  control_restoration / routine_exchange /
↪  mixed_exchange

Do not force all partition-role phases into
↪  the output. Include only the
↪  partition_role labels that meaningfully
↪  shape the set story. Skip phases with
↪  limited evidence or phases that do not
↪  meaningfully affect the set's overall
↪  progression. Keep the selected phases in
↪  chronological order.

Use set_progression to decide:
- which parts of the set shaped the overall
↪  story,
- whether the interval position mattered,
- whether there was a key shift, comeback
↪  attempt, or closing run,
- which phases should be emphasized,
↪  compressed, or omitted in the
↪  final_report,
- which turning points mattered most compared
↪  with nearby events,
- what overall set story the final_report
↪  should emphasize,
- how much narrative attention the set needs.

If the eventual set loser led at an important
↪  point, include that lead when it explains
↪  a comeback, control shift, or failed
↪  advantage.

Keep the evidence short and grounded. Use
↪  score_focus for exact score movement and
↪  tactical_focus for play development
↪  supported by segment_narrative.

For detail_strategy, choose exactly one of
↪  these values:
- brief: use when the set was one-sided or one
↪  player controlled most phases.
- balanced: use when the set was mostly
↪  controlled but included some resistance or
↪  one clear turning point.
- expanded: use when the set was fluctuating,
↪  included a comeback attempt, repeated lead
↪  changes, or a close finish.

# Writing steps:
1. Read set_context and current_report to
↪  understand the set position, match
↪  context, final set score, and continuity
↪  with previous sets.
2. Read set_partition_overview to understand
↪  the compressed full-set arc, then read
↪  reported_segments in chronological order.
3. Build set_progression by analyzing the
↪  important phases of the set in
↪  chronological order.
4. Identify the main turning points and
↪  explain why they mattered more than nearby
↪  events.
5. Use segment_narrative to identify grounded
↪  play development, such as pressure,
↪  control, response, continuity, rally
↪  quality, or error patterns. Do not use it
↪  to recover exact score states because
↪  those are masked as [SCORE].
6. Based on phase_progression,
↪  main_turning_points, and set_story, write
↪  the final_report for the set.

# Input description:
- set_context:
  - set_position: The position of the current
  ↪  set in the match.
  - match_metadata: Match-level contextual
  ↪  information.
    - match_winner: Overall match winner.
    - match_loser: Overall match loser.
    - set_scores: Final scores of match winner
    ↪  and match loser for each set.

- current_report: The report already written
↪  for previous sets in the same match. Exact
↪  score states may be masked as [SCORE]. Use
↪  it only for match continuity and do not
↪  recover exact scores from it.

- set_partition_overview: A chronologically
↪  ordered compressed overview of the full
↪  current set from partition.csv. Use it to
↪  understand the set's overall structure,
↪  especially low-detail or omitted portions
↪  that may not appear in reported_segments.
  Each item contains:
  - partition_no: The chronological partition
  ↪  number inside the set.
  - set_leading: The side leading at the end
  ↪  of the partition.
  - set_trailing: The side trailing at the end
  ↪  of the partition.
  - dominant_role: The most frequent event
  ↪  role inside the partition.
  - role_sequence: The compressed event-role
  ↪  sequence inside the partition.
  - partition_role: The wider phase role of
  ↪  the partition.
  - text_events: Possible compressed
  ↪  phase-level descriptions for the
  ↪  partition. Use these as context only,
  ↪  not as sentences to copy directly.
```

```
- reported_segments: A chronologically ordered
↪  list of reported portions of the current
↪  set.
  Each item contains:
  - segment_id: Segment identifier. Use this
  ↪  only for reference; do not mention it in
  ↪  final_report.
  - segment_type: Intended reporting depth for
  ↪  this portion.
    Possible values:
    - "Low - Omitted"
    - "Moderate - Highlight report"
    - "High - Play-by-play report"

  - reported_segment_position: The position of
  ↪  this reported portion among all reported
  ↪  portions.
  - tension_profile: Ranked signals describing
  ↪  this portion relative to other reported
  ↪  portions.
    Use this only to judge importance and
    ↪  progression.
  - score_events: Important selected score
  ↪  situations inside this reported portion.
  ↪  Use these as the factual backbone for
  ↪  exact score movement. Score states are
  ↪  formatted as
  ↪  match_winner_score-match_loser_score.
    Each score_event contains:
    - score_state: The exact score state.
    - leader: The side leading at that score
    ↪  state, such as match_winner,
    ↪  match_loser, or tied.
    - types: A list of meanings attached to
    ↪  that score state.

    Each type item may contain:
    - label: The score-event meaning, such as
    ↪  final_score, interval_score,
    ↪  equalizer, deuce, early_lead_change,
    ↪  late_lead_change, early_comeback,
    ↪  late_comeback, early_dominance, or
    ↪  late_dominance.
    - description: A short explanation of the
    ↪  score-event meaning.
    - gap_before, gap_after, gap_reduction, or
    ↪  gap_increase when relevant.
    - previous_leader or new_leader when
    ↪  relevant.
  - segment_narrative: The local generated
  ↪  draft for this reported portion. Exact
  ↪  score states from the local draft are
  ↪  masked as [SCORE]. Use score_events, not
  ↪  segment_narrative, as the factual source
  ↪  for exact score states.

# Output:
Return valid JSON only.
{
  "set_progression": {
    "phase_progression": [
      {
        "phase": "opening_contest /
        ↪  opening_dominance /
        ↪  closing_contest /
        ↪  closing_dominance /
        ↪  control_building /
        ↪  control_extension / gap_narrowing
        ↪  / comeback_pressure /
        ↪  contest_reset /
        ↪  control_restoration /
        ↪  routine_exchange /
        ↪  mixed_exchange",
        "phase_summary": "...",
        "leader": "match_winner / match_loser
        ↪  / tied",
        "score_focus": {
          "score_state": "...",
          "score_evidence": "..."
        },
        "tactical_focus": {
          "match_winner_play_development":
          ↪  "...",
         "match_loser_play_development": "..."
        }
      }
    ],
    "main_turning_points": [
      {
        "event_description": "...",
        "why_it_mattered_more_than_nearby_eve
        ↪  nts":
        ↪  "..."
      }
    ],
    "set_story": {
      "overall_arc": "...",
      "detail_strategy": "brief / balanced /
      ↪  expanded",
      "reason": "..."
    }
  },
  "final_report": "..."
}
```

A.4 Match-Level Generation Prompt

```
You are a sports editor-in-chief covering a
↪  badminton match.

# Situation:
You are provided with intermediate reports of
↪  the sets written by sports editors.

# Task:
Write a publishable journalistic report for
↪  the provided badminton match.

Your job is not to compress the set-level
↪  reports further. Instead, use them as
↪  selected evidence and rewrite them into a
↪  coherent match-level article.

# Guidelines:
- Follow the order of events.
```

```
- The final_report must use match_framing as its article plan, repair any local score-ordering problems from the set-level reports, and produce a clear lead, smooth transitions between sets, accurate score progression, and natural journalistic wording.
- Use set_narrative as the main evidence. The set-level reports already contain the selected important details.
- Use score_events to verify and repair score progression. If set_narrative contains out-of-order score mentions, reorder or rewrite the sentence so the final_report follows chronological score movement.
- Use tension_profile only to judge which sets or phases may need stronger emphasis in the match framing.
- Score states in the input are provided in [match_winner_score]-[match_loser_score] format to identify which score belongs to each player. In the final_report, write scores naturally based on the sentence context, making clear which player led, trailed, fought back, forced a decider, or closed out the match.
- Do not further compress the set-level reports. Rewrite them into a publishable article by improving lead, transitions, structure, and journalistic wording.
- Preserve the key score progression from each set, but avoid simply concatenating the set reports.
- Use journalistic interpretation only when it is grounded in both score progression and play development, such as sustained pressure, rally control, attacking response, defensive resistance, forced errors, interval control, comeback pressure, or closing stability.
- In the opening lead of final_report, include the available tournament context from match_metadata, especially tournament, round, date, winner, loser, and duration when provided. Do not repeat these details later unless needed for clarity.
- When rewriting each set, keep score states in chronological order.
- Avoid vague praise or unsupported psychological claims.
- Do not invent facts or make unsupported inferences.
- Do not use planning terms such as "score anchor", "score event", "tension", "point", "sequence", "rank", "set_rewriting_plan", or "match_framing" in the final_report.

# Match framing instructions:
Before writing final_report, identify the journalistic framing of the match.

Use match_framing to decide:
- how the match result should be introduced using tournament, round, date, and duration when available,
- how the sets relate to each other,
- which set or turning point best explains the result,
- what repeated or changing play pattern explains the match result,
- how score progression and play development should be connected across sets,
- how each set should be connected in the article,
- what wording or structure should be smoothed from the set-level reports,
- how the report should end.

Do not use match_framing to remove important set details. Use it to organize and rewrite the selected evidence into a publishable match report.

# Writing steps:
1. Read match_metadata to understand tournament context, players, and match result.
2. Read ordered_sets in chronological order.
3. Build match_framing in the same order as the output schema:
   - Identify lead_angle from the overall match result.
   - Write match_result_summary using match_metadata.
   - Identify set_relationship by comparing how the sets connect.
   - Identify the turning_point that best explains the match result.
   - Identify the decisive_pattern that explains how the winner secured the match.
   - Identify play_development_pattern by comparing how pressure, control, response, rally quality, or error patterns developed across sets.
   - For each set, write set_rewriting_plan to explain the set's role, rewrite focus, wording or structure to smooth, and transition purpose.
   - Write match_headline.
   - Write match_story as a short overview of the full match progression.
   - Write journalistic_style to guide the lead, body, transitions, and ending. The lead_sentence_strategy should specify how to include tournament, round, date, and duration when available.
4. Before writing final_report, check each set for chronological score order. If set_narrative mentions score states out of order, repair the wording using score_events while preserving the selected evidence.
5. Write final_report as one publishable journalistic article that operationalizes every field in match_framing. Use:
   - lead_angle and match_result_summary to shape the opening lead,
   - set_relationship to connect the sets,
   - turning_point to decide the main emphasis,
   - decisive_pattern to explain how the winner secured the match,
   - play_development_pattern to explain the match beyond the scoreline,
   - set_rewriting_plan to rewrite each set with the correct role and transition,
```

```
   - match_headline to guide the article's
   ↪  main angle,
   - match_story to guide the overall
   ↪  progression,
   - journalistic_style to guide the lead,
   ↪  body, transitions, and ending.

# Input description:
- match_metadata:
  - tournament: Tournament name.
  - round: Match round.
  - date: Match date in YYYY-MM-DD format.
  - duration_minutes: Total match duration in
  ↪  minutes.
  - match_winner: Overall match winner.
  - match_loser: Overall match loser.

- ordered_sets: A chronologically ordered list
↪  of set-level reports.
  Each item contains:
  - set_id: Set identifier. Use this only for
  ↪  match_framing.
  - tension_profile: Ranked signals describing
  ↪  this set relative to other sets in the
  ↪  match. Use this only to judge emphasis,
  ↪  not as wording for final_report.
  - score_events: Important selected score
  ↪  situations from this set. Use these to
  ↪  verify and repair exact score
  ↪  progression when rewriting the
  ↪  set_narrative. Score states are
  ↪  formatted as
  ↪  match_winner_score-match_loser_score.
    Each score_event contains:
    - score_state: The exact score state.
    - leader: The side leading at that score
    ↪  state, such as match_winner,
    ↪  match_loser, or tied.
    - types: A list of meanings attached to
    ↪  that score state.

    Each type item may contain:
    - label: The score-event meaning, such as
    ↪  final_score, interval_score,
    ↪  equalizer, deuce, early_lead_change,
    ↪  late_lead_change, early_comeback,
    ↪  late_comeback, early_dominance, or
    ↪  late_dominance.
    - description: A short explanation of the
    ↪  score-event meaning.
    - gap_before, gap_after, gap_reduction, or
    ↪  gap_increase when relevant.
    - previous_leader or new_leader when
    ↪  relevant.
  - set_narrative: The intermediate set-level
  ↪  report. Use this as the main evidence
  ↪  for final_report.

# Output:
Return valid JSON only.
{
  "match_framing": {
    "lead_angle": {
      "lead_label": "straight_games_win /
      ↪  comeback_win / three_set_escape /
      ↪  dominant_win / upset_or_milestone /
      ↪  routine_win / close_finish",
      "evidence": "..."
    },
    "match_result_summary": {
      "winner": "...",
      "loser": "...",
      "scoreline": "...",
      "match_meaning": "..."
    },
    "set_relationship": {
      "set_label": "winner_controlled_all_sets
      ↪  / loser_forced_decider /
      ↪  momentum_shifted_across_sets /
      ↪  close_match_decided_late /
      ↪  winner_recovered_after_set_loss /
      ↪  escalating_contest",
      "evidence": "..."
    },
    "turning_point": {
      "set_id": 1,
      "description": "..."
    },
    "play_development_pattern": {
      "play_label": "sustained_pressure /
      ↪  rally_control / attacking_response /
      ↪  defensive_resistance / forced_errors
      ↪  / interval_control /
      ↪  closing_stability /
      ↪  fluctuating_control",
      "evidence": "...",
      "how_it_connects_the_sets": "..."
    },
    "set_rewriting_plan": [
      {
        "set_id": 1,
        "set_role": "opening_control /
        ↪  response_set / routine_set /
        ↪  decisive_set / comeback_set /
        ↪  closing_set",
        "rewrite_focus": "...",
        "details_to_smooth": ["..."],
        "transition_purpose": "..."
      }
    ],
    "match_headline": "...",
    "match_story": "...",
    "journalistic_style": {
      "lead_sentence_strategy": "...",
      "body_strategy": "...",
      "transition_strategy": "...",
      "ending_strategy": "..."
    }
  },
  "final_report": "..."
}
```

A.5 Flat Match Generation Prompt

```
You are a sports editor-in-chief covering a
↪  badminton match.

# Situation:
You are provided with full match evidence. The
↪  evidence includes match-level context and
↪  set-level details. Your task is to write
↪  the match report directly from this
↪  evidence, without using separately
↪  generated set reports.

# Task:
Write a publishable journalistic report for
↪  the provided badminton match.
```

```
Your job is to build the match story from the set-by-set progression, the important score movements, and the overall match result. Use the match context to frame the article, and use the set evidence to explain how the match unfolded.

# Guidelines:
- Follow the order of events.
- Use match_metadata to understand the tournament context, players, match result, duration, and final scoreline.
- Use ordered_sets in chronological order to understand how each set developed.
- For each set, read set_partition_overview in chronological order.
- Use possible_score_events inside each partition as the main evidence for important score movements and key moments.
- Use each partition's role_sequence, partition_role, dominant_role, and text_events to understand the wider flow of the set.
- Use tension_profile only to decide which sets or phases may need stronger emphasis.
- Use final_score to confirm how each set ended.
- Score states in the input are provided in [match_winner_score]-[match_loser_score] format to identify which score belongs to each player. In the final_report, write scores naturally based on the sentence context, making clear which player led, trailed, fought back, forced a decider, or closed out the match.
- Keep score states in chronological order within each set.
- Preserve the key score progression from each set, but avoid listing events one by one.
- Use journalistic interpretation only when it is grounded in score progression and play development, such as sustained pressure, rally control, attacking response, defensive resistance, forced errors, interval control, comeback pressure, or closing stability.
- In the opening lead of final_report, include the available tournament context from match_metadata, especially tournament, round, date, winner, loser, and duration when provided. Do not repeat these details later unless needed for clarity.
- Avoid vague praise or unsupported psychological claims.
- Do not invent facts or make unsupported inferences.
- Do not copy text_events directly. Use them only as supporting context.
- Do not use planning terms such as "score anchor", "score event", "tension", "point", "sequence", "rank", "set_rewriting_plan", or "match_framing" in the final_report.

# Match story instructions:
Before writing final_report, identify the journalistic framing of the match.

Use match_framing to decide:
- how the match result should be introduced,
- how the sets relate to each other,
- which set or turning point best explains the result,
- what repeated or changing play pattern shaped the match,
- how score progression and play development should be connected across sets,
- how much detail each set needs,
- how the article should end.

Do not use match_framing to remove important set details. Use it to organize the provided evidence into a natural match report.

# Writing steps:
1. Read match_metadata to understand tournament context, players, and match result.
2. Read ordered_sets in chronological order.
3. Build match_framing by identifying:
   - the main lead angle,
   - the match result summary,
   - how the sets connect,
   - the turning point that best explains the result,
   - the decisive pattern across the match,
   - the role of each set in the article,
   - the headline,
   - the full match story,
   - the journalistic style for the lead, body, transitions, and ending.
4. Check the score movement in each set before writing. If any wording would make the score progression unclear, rewrite it so the article follows chronological order.
5. Write final_report as one publishable journalistic article.

# Input description:
- match_metadata: Match-level contextual information.
  - tournament: Tournament name.
  - round: Match round.
  - date: Match date in YYYY-MM-DD format.
  - duration_minutes: Total match duration in minutes.
  - match_winner: Overall match winner.
  - match_loser: Overall match loser.
  - set_scores: Final scores of match winner and match loser for each set.

- ordered_sets: A chronologically ordered list of sets in the match.
  Each item contains:
  - set_id: Set identifier. Use this only for match_framing.
  - final_score: Final score of the set in match_winner_score-match_loser_score format.
  - tension_profile: Ranked signals describing this set relative to other sets in the match. Use this only to judge emphasis, not as wording for final_report.
  - set_partition_overview: A chronologically ordered list of partitions within the set.

    Each partition contains:
```

```
    - partition_no: The chronological
    ↪  partition number inside the set.
    - start_score: Score state at the
    ↪  beginning of the partition.
    - end_score: Score state at the end of the
    ↪  partition.
    - set_leading: The side leading at the end
    ↪  of the partition.
    - set_trailing: The side trailing at the
    ↪  end of the partition.
    - dominant_role: The most frequent event
    ↪  role inside the partition.
    - role_sequence: The compressed event-role
    ↪  movement inside the partition.
    - partition_role: The wider phase role of
    ↪  the partition.
    - text_events: Possible compressed
    ↪  phase-level descriptions for the
    ↪  partition. Use these as context only,
    ↪  not as sentences to copy directly.
    - possible_score_events: Possible
    ↪  important score situations inside this
    ↪  partition. Use these as the main
    ↪  factual evidence for exact score
    ↪  movement.

      Each possible_score_event contains:
      - score_state: The exact score state in
      ↪  match_winner_score-match_loser_sco⌋
      ↪  re
      ↪  format.
      - leader: The side leading at that score
      ↪  state, such as match_winner,
      ↪  match_loser, or tied.
      - event_role: The score-event role, such
      ↪  as final_score, game_point,
      ↪  interval_lead, interval_tie,
      ↪  equalizer, comeback_attempt,
      ↪  dominance_building, or
      ↪  routine_point.
      - event_description: A short factual
      ↪  description of what the score event
      ↪  represents.
      - supporting_details: Additional
      ↪  available details, such as rally
      ↪  position, point winner, gap
      ↪  movement, run length, shot type, or
      ↪  rally-ending reason.

# Output:
Return valid JSON only.

{
  "match_framing": {
    "lead_angle": {
      "lead_label": "straight_games_win /
      ↪  comeback_win / three_set_escape /
      ↪  dominant_win / upset_or_milestone /
      ↪  routine_win / close_finish",
      "evidence": "..."
    },
    "match_result_summary": {
      "winner": "...",
      "loser": "...",
      "scoreline": "...",
      "match_meaning": "..."
    },
    "set_relationship": {
      "set_label": "winner_controlled_all_sets
      ↪  / loser_forced_decider /
      ↪  momentum_shifted_across_sets /
      ↪  close_match_decided_late /
      ↪  winner_recovered_after_set_loss /
      ↪  escalating_contest",
      "evidence": "..."
    },
    "turning_point": {
      "set_id": 1,
      "description": "..."
    },
    "play_development_pattern": {
      "play_label": "sustained_pressure /
      ↪  rally_control / attacking_response /
      ↪  defensive_resistance / forced_errors
      ↪  / interval_control /
      ↪  closing_stability /
      ↪  fluctuating_control",
      "evidence": "...",
      "how_it_connects_the_sets": "..."
    },
    "set_rewriting_plan": [
      {
        "set_id": 1,
        "set_role": "opening_control /
        ↪  response_set / routine_set /
        ↪  decisive_set / comeback_set /
        ↪  closing_set",
        "rewrite_focus": "...",
        "details_to_smooth": ["..."],
        "transition_purpose": "..."
      }
    ],
    "match_headline": "...",
    "match_story": "...",
    "journalistic_style": {
      "lead_sentence_strategy": "...",
      "body_strategy": "...",
      "transition_strategy": "...",
      "ending_strategy": "..."
    }
  },
  "final_report": "..."
}
```