CricBench: A Multilingual Benchmark for Evaluating LLMs in Cricket Analytics

Anonymous ACL submission

Abstract

Cricket is the second most popular sport globally, commanding a massive following of over 2.5 billion fans globally. Enthusiasts and analysts frequently seek advanced statistical insights such as long-term historical performance trends or complex player comparisons that are often unavailable through standard web searches. While Large Language Models (LLMs) have advanced significantly in Text-to-SQL tasks, their capability to handle the domain-specific nuances, complex schema variations, and multilingual requirements inherent to sports analytics remains under-explored. To investigate this potential capability gap, we present **CricBench**, a comprehensive benchmark suite for evaluating LLMs on specialized cricket data. To curate a "Gold Standard" dataset, we collaborate with domain experts in cricket and SQL to manually author complex queries, ensuring logical correctness. Recognizing linguistic diversity, we construct the benchmark in both English and Hindi, establishing a framework that is open for further extension to other regional languages. We evaluate six state-of-the-art models—including GPT-4o, Claude 3.7 Sonnet, and open-source models using a strict evaluation protocol. Our results reveal that high performance on general benchmarks does not guarantee success in specialized domains. While the open-weights reasoning model **DeepSeek R1** achieves state-of-the-art performance (50.6%), surpassing proprietary giants like Claude 3.7 Sonnet (47.7%) and GPT-4o (33.7%), it still exhibits a significant accuracy drop when moving from general benchmarks (BIRD) to CricBench. Furthermore, we observe that code-mixed Hindi queries frequently yield parity or higher accuracy compared to English, challenging the assumption that English is the optimal prompt language for specialized SQL tasks.

1 Introduction

Large Language Models (LLMs) have achieved significant milestones in translating natural language into structured query language (Text-to-SQL), fundamentally altering how non-technical users interact with databases (Chen et al., 2021; Achiam et al., 2023). Pioneering benchmarks such as WikiSQL (Zhong et al., 2017) and Spider (Yu et al., 2018) have driven rapid improvements in schema linking and syntactic correctness (Wang et al., 2020). More recently, the BIRD benchmark (Li et al., 2023) has pushed the frontier by integrating external knowledge reasoning. However, while general-purpose models excel at standard business queries, the extent to which these capabilities transfer to highly specialized domains remains an open research question (Suhr et al., 2020).

Sports analytics, particularly cricket, represents a unique testbed for investigating this "domain gap." Cricket is a global phenomenon with over 2.5 billion fans, yet it remains under-represented in computational linguistics research. Unlike general knowledge questions, this domain necessitates granular reasoning involving precise mathematical operations and temporal awareness. This includes the computation of compound derived metrics and the application of phase-specific filters. Furthermore, the domain presents significant entity resolution challenges, such as reconciling evolving nomenclature for organizations and resolving dynamic player affiliations across varying temporal contexts, thereby testing an LLM's ability to maintain knowledge consistency (Wei et al., 2022).

Current Text-to-SQL benchmarks are predominantly English-centric (Yu et al., 2018; Li et al., 2023). This limitation is particularly critical for sports like cricket, where the fanbase is linguistically diverse. In regions like India, users often interact in native languages or code-mixed dialects. A model that generates accurate SQL for an English query but fails to process the same intent in Hindi is of limited utility in real-world deployment (He et al., 2019).

To address these challenges, we introduce **CricBench**, a specialized, multilingual benchmark designed to evaluate the robustness of LLMs in cricket analytics. CricBench provides a rigorous testing ground by combining a normalized database schema with a set of high-difficulty questions in English and Hindi. Unlike synthetic datasets generated via automation, CricBench is curated through a **collaborative expert-driven process**. We engaged domain experts in cricket analytics and SQL engineering to manually author and verify queries, ensuring that the benchmark reflects authentic reasoning patterns rather than artifacts

of model generation. We evaluate a suite of frontier models, including GPT-4o (Achiam et al., 2023), Claude 3.7 Sonnet (Anthropic, 2024), and open-source contenders like Llama 3 (AI, 2024) and Qwen 2.5 (Group, 2024), to provide a comprehensive assessment of the current state of AI in sports intelligence. Our primary contributions are as follows:

- **CricBench Dataset:** We release the first expert-curated, domain-specific Text-to-SQL benchmark for cricket analytics, consisting of normalized schemas and high-complexity queries.[1]
- **Context-Aware Evaluation:** We propose a dynamic complexity router to mitigate knowledge gaps in LLMs. This module injects critical domain context such as historical team mappings and match-phase logic, ensuring the evaluation captures nuances that standard schema-linking misses.
- **Multilingual Support:** We introduce a verified Hindi test set that incorporates authentic code-mixing, addressing the lack of Indic language resources in specialized SQL benchmarks.
- **SOTA Analysis:** We identify DeepSeek R1 as the current state-of-the-art for this domain (50.6% accuracy), outperforming GPT-4o and Claude 3.7, establishing a new baseline for open-weights reasoning in SQL generation.
- **Benchmarking & Analysis:** We evaluate six state-of-the-art LLMs in both **Raw (Zero-Shot)** and **Context-Aware** settings. Our analysis reveals that the Router is transformative for frontier models (boosting GPT-4o's accuracy by 72%), whereas generalist models struggle with the domain gap, suffering significant degradation compared to general-purpose benchmarks like BIRD.
- **Expert Curation:** Unlike recent trends in synthetic data generation, our Gold Standard is manually authored and verified by domain experts to ensure logical and factual correctness.

2 Related Work

The development of Natural Language Interfaces to Databases (NLIDB) has been a longstanding goal in NLP. Our work builds upon and extends research in three key areas: Text-to-SQL generation, domain-specific benchmarking, and multilingual evaluation.

2.1 Text-to-SQL Benchmarks

The standard for evaluating Text-to-SQL systems has historically been set by large-scale, cross-domain datasets. WikiSQL (Zhong et al., 2017) provided a foundation for simple SELECT-based queries, while Spider (Yu et al., 2018) introduced complex, multi-table joins and nested queries, becoming the de facto benchmark for measuring generalization across unseen domains. More recently, BIRD (Li et al., 2023) has pushed the frontier by focusing on "dirty" database contents and external knowledge reasoning. While these benchmarks are invaluable for general-purpose evaluation, they often lack the depth required for intense domain specificity found in specialized fields like sports analytics.

2.2 Domain-Specific Query Generation

Research has increasingly demonstrated that generalist models often struggle with the nuances of specialized domains like healthcare and finance, where "correct" SQL generation depends heavily on understanding domain logic rather than just schema linking (Suhr et al., 2020). Previous works have attempted to mitigate this via Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) and few-shot prompting (Brown et al., 2020). However, standard retrieval often misses strict logical constraints (e.g., specific calculation formulas). Our work extends this by implementing a deterministic routing layer. Instead of retrieving unstructured context, our system injects precise domain rules (e.g., franchise evolution, match phases) to fill the parametric knowledge gap, ensuring the model can handle details that were not present in its initial training data. Additionally, we employ a rigorous manual authoring process by domain experts (Wei et al., 2022) to ensure high data fidelity.

2.3 Multilingual Evaluation

While English-centric Text-to-SQL research is mature, multilingual benchmarks remain scarce for Indic languages. Existing resources often focus on high-resource languages or rely on machine-translated versions of general benchmarks (He et al., 2019). Our work addresses this gap by introducing a verified Hindi test set. We specifically address the challenge of code-mixing, where technical terms are retained in English (e.g., "Strike Rate") within Hindi sentence structures, mirroring authentic user behavior in the Indian subcontinent.

3 Methodology

CricBench is a benchmark suite encompassing data engineering, expert curation, and a specialized benchmarking pipeline. The system architecture is designed to test not just SQL generation, but also domain-specific reasoning and multilingual understanding.

3.1 Dataset Construction

The foundation of CricBench is a robust, normalized relational database. We ingested comprehensive ball-by-ball records sourced from Cricsheet (Cricsheet, 2024), transforming semi-structured JSON data into a strict SQLite schema (Hipp, 2024). The database consists of five core tables:

- **Matches:** Metadata for each game, including dates, venues, and results. Chronology is strictly enforced via `match_date`.

[1] The dataset and code will be made publicly available upon acceptance.

- **Deliveries:** Granular ball-by-ball events, serving as the source of truth for all derived statistics.
- **Players:** A central registry for entity resolution (`player_id`, `player_name`).
- **PlayerInMatch:** A linking table associating players with specific matches and teams.
- **FielderDismissals:** A specialized junction table linking dismissal events to specific fielders.

3.2 Gold Standard Curation

To ensure the benchmark's reliability, we eschewed automated generation in favor of a rigorous **manual curation process**.

3.2.1 Expert Authoring & Verification

We collaborated with domain experts in cricket analytics to manually author 200 high-complexity natural language questions. These inquiries were derived directly from real-world analysis patterns observed in match commentaries and statistical leaderboards on platforms like ESPNcricinfo and Cricbuzz.

To illustrate the "Domain Gap" our dataset targets, Table 1 presents a representative Code-Mixed Hindi query. It demonstrates the necessity of filtering for specific game phases ("Death Overs") and team affiliations, a nuance where generalist models like **GPT-4o (Zero-Shot)** often fail, while specialized reasoning models like **DeepSeek R1** succeed.

Query: *"Bumrah ka death overs mein Economy Rate kya hai Mumbai Indians ke liye?"* **Intent:** Calculate Economy Rate (Overs 16-20) for Bumrah playing for Mumbai Indians.
DeepSeek R1 (Success): Correctly identifies "Death Overs" as `over_number > 15`, joins `PlayerInMatch` for team filtering, and applies the correct economy formula.
GPT-4o (Baseline Failure): Hallucinates a `total_runs` column, fails to filter for "Legal Balls" (wides/no-balls), and ignores the death over constraint entirely.

Table 1: Illustrative Case Study: Handling specialized domain logic in Hindi code-mixed queries.

3.2.2 Difficulty Taxonomy & Distribution

To quantify the complexity of CricBench, we categorize queries based on required SQL features and domain-specific challenges. As shown in Table 2, the benchmark leans heavily towards complex, multi-hop reasoning rather than simple retrieval.
We define these complexity categories as follows:

- **Structural Complexity:** Queries requiring the joining of multiple normalized tables (e.g., `PlayerInMatch` → `Matches`) or nested logic (e.g., finding a player's rank within a filtered subset).
- **Franchise Normalization:** Queries targeting teams that have undergone name changes (e.g., *Delhi Daredevils* → *Delhi Capitals*), requiring the model to inject conditional `CASE` logic to aggregate records correctly across eras.

Complexity Category	Count	% of Dataset
Multi-table Joins	150	75.0%
Nested Queries / CTEs	85	42.5%
Aggregations (Group By/Having)	170	85.0%
Temporal Filtering	119	59.5%
Franchise Normalization	79	39.5%
Derived Metrics (SR, Econ)	43	21.5%

Table 2: Distribution of query complexity in CricBench. Categories are not mutually exclusive; a single query often involves joins, temporal logic, and normalization.

- **Temporal Reasoning:** Questions dependent on strict chronological ordering (e.g., *"Last 5 matches"*, *"Debut match"*), which force the model to resolve `match_date` rather than relying on ID sorting.
- **Derived Metrics:** Queries requiring mathematical composition of raw columns, such as calculating *Economy Rate* from runs conceded and legal balls bowled.

Verification: For every question, the corresponding SQL query was hand-written. The answers were manually cross-referenced against authentic third-party records from Cricbuzz (Cricbuzz, 2024) and ESPNcricinfo (ESPN Cricinfo, 2024) to guarantee factual correctness.

Multilingual Expansion: The dataset was manually translated into Hindi, specifically preserving technical terms via code-mixing (e.g., retaining "Strike Rate" in English script) to mirror authentic user behavior.

3.3 Benchmarking Framework

To rigorously quantify the value of domain context, we implemented a dual-mode evaluation framework. This allows us to isolate the model's intrinsic SQL capabilities from its ability to follow complex domain instructions.

3.3.1 Experimental Configurations

We evaluate all models under two distinct settings:

1. **Raw Evaluation (Baseline):** The model is provided **only** with the basic database schema (table names, columns, and foreign keys) and the natural language question. No definitions for metrics (e.g., Economy Rate) or team mappings are provided. This tests the model's zero-shot ability to generalize from parametric knowledge.

2. **Context-Aware Evaluation (Ours):** The model receives the full output of our **Complexity Router**, which includes a Global Preamble (metric definitions) and dynamically injected context rules (team mappings). This tests the model's ability to adhere to strict domain constraints.

3.3.2 Complexity Routing Algorithm

For the Context-Aware setting, we employ a heuristic router to analyze the input query and determine the nec-

essary context injection. To bridge the gap between general-purpose pre-training and domain-specific requirements, this deterministic module functions as a pre-processing layer that scans input queries for specific semantic triggers—such as historical franchise names or tournament phases.

Our router uses a keyword-based heuristic to classify queries into "Context Buckets." Based on this classification, the system dynamically assembles the prompt, injecting precise SQL mapping rules only when relevant. This strategy prevents "Context Pollution"—where irrelevant instructions degrade model performance—while ensuring that critical logic (like normalizing team names) is strictly enforced when needed. This logic is formalized in Algorithm 1.

Algorithm 1 Complexity-Based Context Injection

Input: User Query Q
Output: System Prompt Context C
Define $K_{franchise} \leftarrow$ {'Delhi', 'Punjab', 'Deccan', ...}
Define $K_{playoff} \leftarrow$ {'Qualifier', 'Eliminator', 'Final'}
if $\exists k \in K_{franchise}$ such that $k \in Q$ **then**
    $C \leftarrow$ INJECT_FRANCHISE_MAPPINGS
end if
if $\exists k \in K_{playoff}$ such that $k \in Q$ **then**
    $C \leftarrow$ INJECT_MATCH_TYPE_FILTER
end if
if $\neg\exists k \in K_{franchise} \cup K_{playoff}$ such that $k \in Q$ **then**
    $C \leftarrow$ GLOBAL_PREAMBLE_ONLY
end if
return C

3.3.3 System Prompt Engineering

Based on the router's output, we construct the system prompt using a standardized "Global Preamble" augmented by dynamic rules.

1. Global Preamble (Injected for ALL Queries in Context-Aware Mode): This block establishes the "Imperative Rules" for calculation to prevent common logic errors.

> **CRITICAL CALCULATION RULES:**
>
> 1. **Legal Deliveries:** Count only balls where `wides=0 AND noballs=0`.
> 2. **Batsman Runs:** Use `SUM(runs_scored)`.
> 3. **Team Totals:** Use `SUM(runs_scored + extra_runs)`.
> 4. **Debut Date:** Always use `MIN(match_date)`, NEVER `match_id`.
> 5. **Result Limiting:** If 'top 5', MUST use `LIMIT 5`; if 'highest', MUST use `LIMIT 1`.

2. Dynamic Context Rules (Example): When the router detects historical franchise names (e.g., "Kings XI Punjab"), the following mapping rule is injected to enforce entity normalization:

> **TEAM MAPPINGS:** You MUST use a `CASE` statement to group historical names:
>
> - 'Kings XI Punjab', 'Punjab Kings' → 'Punjab Kings'
> - 'Delhi Daredevils', 'Delhi Capitals' → 'Delhi Capitals'

3. Constraint Injection: Finally, we inject a "Critical Constraint" forcing the model to return result sets with specific column names and sorting orders that match the Gold Standard for automated evaluation.

3.4 Evaluation Metrics

We evaluate model performance using three primary metrics to distinguish between syntactic correctness, instruction following, and semantic understanding:

- **Execution Accuracy (EX):** Measures whether the generated SQL query executes against the database without throwing syntax or logic errors.
- **Schema Compliance:** Verifies that the model's output schema (column names and order) aligns exactly with the Gold Standard definition.
  Justification for Strictness: While standard Text-to-SQL tasks often permit flexible aliases, our benchmark explicitly injects schema constraints into the system prompt (see Appendix A). This metric therefore serves as a strict test of **instruction following** and ensures compatibility with downstream applications that expect deterministic output formats.
- **Data Match Accuracy (Dynamic Tolerance):** Comparing numerical outputs in sports analytics requires tolerance for floating-point precision issues. We implemented a context-aware comparison protocol:
  - **Type Identification:** Tolerance thresholds of ± 1.0 are applied based on the Gold Standard column headers.
  - **Granularity:** The check is performed **cell-by-cell**. A generated row is considered a "match" only if *every* numerical cell within that row falls within its specific tolerance threshold and all string cells match exactly.

4 Results and Discussion

4.1 Quantitative Analysis

We evaluated models in two configurations: **Raw** (standard zero-shot prompting) and **Context-Aware** (using our Complexity Router and Global Preamble). Table 3 presents the comparative performance.

4.2 The Router Effect: Universal Utility vs. Formatting Cost

Our experiments demonstrate that injecting domain context is beneficial for nearly all architectures, but it creates different dynamics based on model capacity.

Model	Raw (Baseline)			Context-Aware (Router)			Δ Match
	Exec	Schema	Match	Exec	Schema	Match	
DeepSeek R1	96.2%	95.0%	42.6%	**98.1%**	**96.7%**	**50.6%**	+8.0%
Claude 3.7 Sonnet	82.6%	82.2%	40.5%	87.9%	87.4%	47.7%	+7.2%
GPT-4o	88.5%	86.8%	19.6%	88.5%	86.0%	33.7%	**+14.1%**
Qwen 2.5 (7B)	56.3%	35.6%	2.9%	52.7%	21.5%	4.8%	+1.9%
Llama 3.1 (8B)	53.1%	43.1%	2.3%	64.2%	23.2%	2.8%	+0.5%
Gemma 2 (9B)	65.9%	44.4%	5.7%	70.1%	22.6%	3.9%	-1.8%

Table 3: Comparison of Raw vs. Context-Aware performance. The Router improves Data Match accuracy for 5 out of 6 models. *Match* denotes strict row-level data accuracy. Note the massive gain for GPT-4o (+14.1%) and the formatting regression (Schema %) for small models.

Frontier Models For high-capacity models, the Complexity Router is transformative. **GPT-4o** exhibited the most dramatic improvement, jumping from 19.6% accuracy in the Raw setting to 33.7% with the Router—a **72% relative improvement**. In the Raw setting, GPT-4o frequently generated syntactically correct SQL (88.5% Exec) that returned incorrect data (19.6% Match). The Router successfully bridged this gap by injecting precise mathematical definitions for domain related metrics.

Small Models For smaller open-weights models (7B-9B), the Router improved logical reasoning but degraded formatting adherence. **Qwen 2.5 (7B)** saw its data accuracy nearly double (2.9% $\rightarrow$ 4.8%) and **Llama 3.1** also improved. However, the increased prompt length caused a drop in **Schema Compliance** (e.g., Qwen dropped from 35.6% to 21.5%). This suggests that 7B-parameter models can leverage the definitions to retrieve the correct data, but the heavy context distracts them from strict output constraints (e.g., specific column aliasing).

4.3 Performance of DeepSeek R1

A critical finding of this study is the dominance of the open-weights model **DeepSeek R1**. It achieved the highest accuracy across all metrics, reaching **50.6%** Data Match, effectively establishing a new state-of-the-art for this domain. Unlike standard instruction-tuned models which often rely heavily on schema-linking aids, DeepSeek R1 demonstrated superior intrinsic reasoning capabilities. This is most evident in the raw evaluation setting, where it maintained a high accuracy of **42.6%** on complex nested queries, significantly outperforming GPT-4o, which achieved only **19.6%** in the same setting. This result suggests that specialized reasoning models may be bridging the "domain gap" more effectively than larger general-purpose proprietary models.

4.4 The Domain Gap: General vs. Specialized

To contextualize our findings, we benchmarked the same suite of models against the BIRD development set ($N = 1534$), a leading general-purpose Text-to-SQL benchmark. Table 4 illustrates the performance gap.

This comparison yields a striking insight into **Semantic Robustness**. Both **DeepSeek R1** and **Claude 3.7** exhibit minimal degradation ($< 5\%$ drop) when moving from general to specialized tasks. In contrast, **GPT-4o** suffers a significant drop (-21.7%), indicating that while it excels at standard schema linking, it struggles to integrate rigid, non-standard definitions required for sports analytics without heavy prompting.

4.5 Multilingual Capabilities

We observed a "Hindi Advantage" across top-tier models. For DeepSeek R1 (Router), the accuracy on the second Hindi test set (**55.0%**) exceeded its English performance (**51.9%**). This validates our code-mixing strategy: by retaining technical keywords (e.g., "Powerplay", "Wicket") in English script within Hindi grammar, we effectively ground the model's reasoning while reducing ambiguity often found in purely English natural language queries.

5 Error Analysis

To understand the underlying causes of the domain gap, we conducted a forensic analysis of the failed queries in the Context-Aware setting. Table 5 presents a comparative breakdown.

5.1 The "Illusion of Competence"

High execution rates can be misleading. In the Raw setting, GPT-4o achieved 88.5% Execution Success but only 19.6% Data Match. This indicates an "Illusion of Competence," where the model generates valid SQL that runs without error but returns factually incorrect data (e.g., hallucinates `total_runs` columns). The Router reduced Schema Hallucination errors to just 4.7% for GPT-4o, but the model still struggled with **Row Count Mismatch** (50% of failures), frequently failing to apply correct `LIMIT` or `RANK()` logic for tie-breaking.

5.2 Context Overload in Small Models

The forensic data confirms the trade-off hypothesis for small models. **Gemma 2** suffered 222 Schema Mismatch errors in the Context-Aware setting (48% of its failures). Qualitative inspection reveals that when presented with the lengthy Preamble, the model often ignored the output constraints, inventing columns like `match_year` or `winner_id` that do not exist in the provided schema, despite having access to the definitions.

Model	BIRD (Gen)	CricBench (Spec)	Domain Gap
DeepSeek R1	55.0%	**50.6%**	**-4.4%**
Claude 3.7	51.7%	47.7%	-4.0%
GPT-4o	**55.4%**	33.7%	-21.7%
Qwen 2.5 (7B)	42.4%	4.8%	-37.6%
Gemma 2 (9B)	38.3%	3.9%	-34.4%
Llama 3.1 (8B)	39.7%	2.8%	-36.9%

Table 4: The Domain Gap: Comparison of accuracy on BIRD (Li et al., 2023) vs. CricBench (Context-Aware). A small negative gap indicates high robustness to specialized domain logic.

Failure Category	DeepSeek R1	Claude 3.7	GPT-4o	Gemma 2	Qwen 2.5	Llama 3.1
Total Failures (N)	*236*	*250*	*317*	*459*	*455*	*464*
1. Syntax / Exec Error	**9 (3%)**	58 (23%)	55 (17%)	143 (31%)	226 (49%)	171 (37%)
2. Schema Mismatch	6 (2%)	2 (1%)	9 (3%)	**222 (48%)**	139 (30%)	191 (41%)
3. Row Count Mismatch	128 (54%)	91 (36%)	**158 (50%)**	73 (16%)	57 (12%)	69 (15%)
4. Value Mismatch	128 (54%)	135 (54%)	120 (38%)	23 (5%)	38 (8%)	38 (8%)

Table 5: Distribution of failure modes. Note: DeepSeek R1 has near-perfect execution (only 9 syntax errors), whereas Gemma 2 suffers massive schema hallucination (222 errors).

6 Limitations

CricBench is currently limited to IPL data and static historical records; it does not yet support real-time live match queries. Additionally, the evaluation of Indic languages is limited to Hindi, excluding other major languages like Tamil or Bengali.

Acknowledgments

We acknowledge the use of Gemini (Google DeepMind) for assistance in enhancing the grammatical precision and presentation of this paper. We certify that the experimental design, data curation, analysis, and all scientific conclusions presented herein accurately represent the original contributions of the authors.

7 Data Availability

To facilitate future research in domain-specific Text-to-SQL, we will make all artifacts from this study publicly available upon acceptance.

Released Assets The release package will include:

1. **CricBench Database:** The fully normalized SQLite database (`ipl.db`) containing granular ball-by-ball data for 1,169 matches (2008–2024).
2. **Gold Standard Dataset:** The expert-verified dataset (`.json`) containing 200 pairs of natural language queries (English & Hindi) and their corresponding ground-truth SQL.
3. **Evaluation Harness:** The complete Python codebase for the Context-Aware Router, the execution engine, and the metric calculation scripts (Execution Accuracy, Schema Compliance, and Data Match).

8 Conclusion

This study introduces **CricBench**, evaluating the readiness of state-of-the-art LLMs for specialized sports analytics. Our findings quantify a significant "domain gap" between general-purpose SQL generation and the nuanced requirements of cricket data. While models like GPT-4o and Claude 3.7 excel on broad benchmarks such as BIRD, they exhibit a precipitous drop in performance when tasked with the granular reasoning required for cricket metrics. Although the open-weights model DeepSeek R1 establishes a new state-of-the-art with 50.6% accuracy, this ceiling remains considerably lower than typical results in general domains, underscoring that current frontier models still lack the intrinsic specialized reasoning necessary for highly technical verticals.

Furthermore, our analysis of context-aware generation reveals critical limitations in current retrieval-augmented strategies. While providing explicit schema definitions and domain rules—via our complexity router—did yield performance gains, the absolute accuracy remains modest across all models. Even when armed with comprehensive context, LLMs frequently struggled to synthesize complex, multi-step constraints correctly. This suggests that the challenge of sports analytics cannot be solved solely through prompt engineering or context injection; rather, it requires fundamental advancements in how models internalize and reason over domain-specific logic.

9 Ethics Statement

All data used in CricBench is derived from publicly available sports statistics. No personal private information (PII) of players or fans is included. The dataset

is intended solely for research purposes to advance the field of specialized Text-to-SQL generation.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Meta AI. 2024. The llama 3 herd of models. *arXiv preprint*.

Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. Technical report, Anthropic.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*.

Mark Chen, Jerry Tworek, Heewoo Jun, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Cricbuzz. 2024. Ipl scores, coverage & statistics. https://www.cricbuzz.com.

Cricsheet. 2024. Ball-by-ball cricket data. https://cricsheet.org.

ESPN Cricinfo. 2024. Ipl stats, coverage & records. https://www.espncricinfo.com.

Alibaba Group. 2024. Qwen2.5 technical report.

Pengcheng He and 1 others. 2019. Xsql: Reinforce schema representation with context. *arXiv preprint arXiv:1908.08113*.

D. Richard Hipp. 2024. SQLite. https://www.sqlite.org.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, and Tim Rocktäschel. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*.

Jinyang Li, Binyuan Hui, Reynold Cheng, Bowen Qin, Chenhao Ma, Nan Huo, Fei Huang, Wenyu Du, Luo Si, and Yongbin Li. 2023. Bird: Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. In *Advances in Neural Information Processing Systems*.

Alane Suhr and 1 others. 2020. On the generalization of text-to-sql models to unseen domains. *COLING*.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. In *ACL*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shan Roman, and 1 others. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. In *Salesforce Research*.

A Appendix: Prompting Specifications

Our system utilizes a dynamic prompting strategy. All queries receive the **Global Preamble**, while specific queries receive additional **Context Injections** based on routing logic and schema requirements.

A.1 Global Preamble (System Prompt)

The following system prompt is prepended to every API call. Crucially, it provides the mathematical definitions for derived statistics to ensure consistent calculations:

Global Preamble

```
You are an expert SQL writer for an IPL cricket database.
IMPERATIVE RULE 1 (Result Limiting):
- If user asks for specific number (e.g. 'top 5'), MUST use LIMIT 5.
- If user asks for absolute rank (e.g. 'highest'), MUST use LIMIT 1.
- For all other queries, return ALL matching rows.
CRITICAL CALCULATION RULES:
1. Legal Deliveries: Count only balls where wides=0 AND noballs=0.
2. Batsman Runs: Use SUM(runs_scored).
3. Team Totals: Use SUM(runs_scored + extra_runs).
4. Debut Date: Always use MIN(match_date), NEVER match_id.
5. Bowler's Wickets: Filter WHERE wicket_type IS NOT NULL AND wicket_type NOT IN ('run out', 'retired hurt').
STATISTICAL DEFINITIONS:
- Bowler Economy: (Runs Conceded * 6.0) / Legal Balls.
- Strike Rate: (Runs Scored * 100.0) / Balls Faced (excluding wides).
- Overs Bowled: Legal Balls / 6.0.
```

A.2 Dynamic Context Injection

1. Team Entity Resolution: If the router detects team names (e.g., "DD", "Kings XI"), the following mapping rule is injected:

Team Mappings

```
- IMPERATIVE Team Franchise Mappings: You MUST use a CASE statement to group historical names.
- Delhi Franchise: Group 'Delhi Daredevils', 'Delhi Capitals' → 'Delhi Capitals'
- Punjab Franchise: Group 'Kings XI Punjab', 'Punjab Kings' → 'Punjab Kings'
... [others omitted for brevity]
```

2. Output Schema Constraint: To enable automated evaluation, we inject a strict constraint ensuring the output columns match our Gold Standard:

Critical Constraint

```
CRITICAL CONSTRAINT: The final result set MUST have exactly these columns in this specific order: [column_list]. Use SQL ALIASES (AS ...) to match these names exactly.
```