Cards Against Contamination: TCG-Bench for Difficulty-Scalable Multilingual LLM Reasoning

Anonymous EACL submission

Abstract

Benchmarks for language models have become essential tools for research. Yet, such benchmarks face a persistent contamination problem, with recent studies finding 25–50% of evaluation datasets appearing in training corpora. This is true even looking at the two-player zero-sum game setting, where most benchmarks are based on popular games, like chess, whose optimal strategies are all over the web. Such contamination hinders the possibility to differentiate memorization and reasoning skills. To rectify these problems, we introduce *TCG-Bench*, a benchmark based on a new two-player trading card game (TCG), similar in spirit to games like *Magic: The Gathering*. TCG-Bench offers three key innovations: (1) a contamination-resistant design by separating the publicly released game engine from hidden card implementations, (2) a continuous difficulty spectrum via Monte Carlo simulation that prevents benchmark saturation, and (3) a parallel implementation in English and Arabic, the first multilingual text-based game benchmark to do so. We also formalize a practical threat model and refresh protocol that preserves evaluation integrity even if specific cards leak. Our analysis across 17 models (50,000+ games) reveals that performance declines exponentially with difficulty, while model size correlates only weakly with strategic ability. We also observe cross-linguistic performance gaps between English and Arabic, with a gap of 47.4% at 32B, highlighting the need for multilingual game benchmarks that target reasoning capabilities in the target language. We host a leaderboard showcasing these results and welcome evaluation requests on our private cards.

1 Introduction

Modern research on language models critically depends on the availability of high-quality benchmarks to measure capabilities. However, the reliability of language model evaluation has become increasingly uncertain as more benchmark data appear in training corpora. Recent analyses reveal concerning levels of contamination: 29.1% for MMLU (Deng et al., 2024), 28.7% for ARC-Challenge (Deng et al., 2024), 25% for HumanEval (Yang et al., 2023), and over 50% for TriviaQA (Madaan et al., 2024). This contamination significantly distorts performance metrics, with models scoring up to 44.5 percentage points higher on contaminated subsets (Yang et al., 2023; OpenAI, 2023). Perhaps most concerning, models such as GPT-4 can identify the correct answer in MMLU even when options are masked (Deng et al., 2024), suggesting a reliance on memorization rather than reasoning.

Why games for evaluation? Two-player games naturally induce partial information, non-myopic trade-offs, and adversarial dynamics—all properties that stress test planning and reasoning beyond surface pattern matching. Unfortunately, reusing *existing* games invites contamination because strategies and walkthroughs are extensively documented and scraped into pretraining corpora. As training data scales, time-separation strategies quickly become obsolete. Dynamic benchmarks introduce variability and maintenance burden. Private evaluations limit participation. Synthetic data often lacks real-world nuance. These challenges invite a different approach: protect evaluation *structurally*, not with secrecy alone.

Our approach. We present TCG-Bench, a text-based trading card game benchmark designed to reduce data contamination by separating a public game engine from private card implementations. It also includes a tunable difficulty controller and a bilingual (English/Arabic) setup, enabling flexible and cross-lingual evaluation of model performance. The system architecture is described in Figure 1.

Our main contributions are:

(i) A contamination-resistant benchmark: Engine/content split with a public leaderboard and private card implementations.

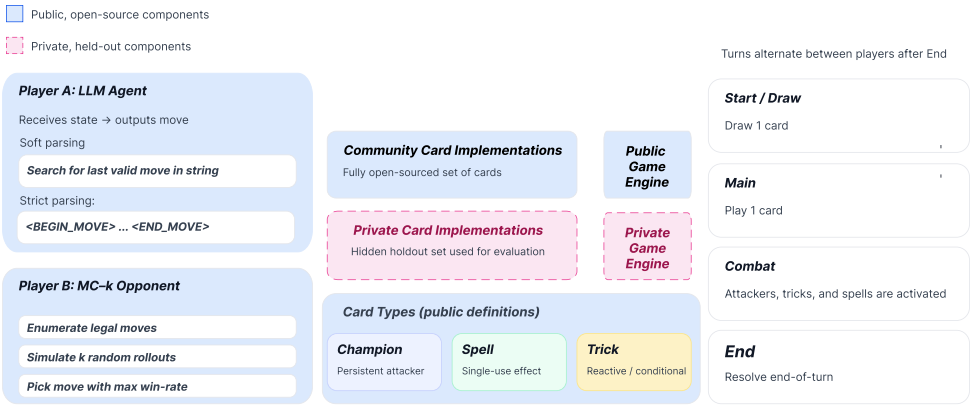

Figure 1: TCG-Bench's architecture.

(ii) Tunable difficulty: Monte Carlo rollouts create a continuous opponent-strength spectrum that remains challenging as models improve.

(iii) Bilingual design: Parallel English and Arabic implementations enable direct cross-linguistic comparisons.

(iv) Threat model & refresh protocol: A practical model of potential leaks with an efficient path to refresh the private content without redesigning the benchmark.

(v) A comprehensive evaluation of 17 models on a variety of axes. Empirically, we find (1) exponential performance decay as difficulty increases ($R^2 > 0.93$); (2) a weak size/performance correlation ($r = 0.31$) with family-level counterexamples; (3) widening English–Arabic gaps with model scale (up to 47.4% at 32B model). Moreover, we also show how the contamination effect is real by showing a big performance gain with a fine-tuned model on games data.

2 Related Work

Mitigating Benchmark Contamination. Early alarms about data leakage in GLUE and SuperGLUE prompted post-hoc filtering and dynamic datasets (Jacovi et al., 2023; Deng et al., 2024). Detection methods have evolved from n-gram overlap (Yang et al., 2023) to embedding-based similarity (Madaan et al., 2024) and guided prompting that reveals memorization (Deng et al., 2024). These post-hoc approaches, however, require access to training data and offer only retrospective diagnosis rather than prevention.

Temporal strategies attempt to stay ahead of contamination by using recent data. LiveBench (White et al., 2025) and LatestEval (Li et al., 2024) refresh items monthly from recent sources, assuming a lag between data creation and training corpus inclusion. This approach faces two challenges. First, the temporal gap is shrinking as models incorporate more recent web snapshots and near-real-time sources. Second, frequent regeneration introduces uncontrolled variability that complicates longitudinal comparison. Dynamic generation (Perez et al., 2022) via LLMs offers another path but risks homogeneity and artifacts from the generator's biases.

Private evaluation sidesteps public exposure entirely. Closed benchmarks like LMSYS Chatbot Arena (Zheng et al., 2023) and proprietary industry suites maintain integrity through restricted access but limit community participation and reproducibility. TCG-Bench differs by enforcing isolation through an engine/content split, reducing the need for continual item renewal while allowing complex, dynamic interactions without variability from regeneration. Our approach offers a middle ground: the engine remains open for transparency and extensibility, while only the card implementations stay private and modular for low-cost refresh.

Game-Based Evaluation of Language Models. Text-adventure frameworks such as Jericho (Côté et al., 2019) and TextWorld (Côté et al., 2018) test language understanding and planning through interactive fiction. While these environments offer rich natural language interaction and partial observability, their public game files are vulnerable to memorization. We verified this risk by detecting Zork and other Jericho games in the C4 corpus (Raffel et al., 2020), we accomplish this by running an Elastic Search service on the entire corpus against the Jericho games through a fuzzy search. TextWorld mitigates this through procedural generation but at the cost of limited narrative depth and stereotyped puzzles, yet still we find traces of its entire

Github repository The Stack (Kocetkov et al., 2022) through the deployed service "Am I in The Stack".

Board and strategy games offer another evaluation paradigm. Recent work has explored chess (Ruoss et al., 2024), Go (Guo et al., 2024), Diplomacy (Bakhtin et al., 2022), and multi-agent negotiation games (Park et al., 2023). However, established games suffer from extensive documentation of optimal strategies and game databases that appear in training corpora.

Surveys (Yang et al., 2024) highlight growing interest in games as LLM benchmarks. GameBench (Costarelli et al., 2024) uses multiple existing games to probe strategic reasoning. TextArena (Guertler et al., 2025) benchmarks agentic reasoning in competitive text games. TCG-Bench is designed for contamination resilience, achieved through a clean architectural separation between components and a modular mechanism for continuous content refresh.

Multilingual Reasoning Benchmarks. Recent efforts probe reasoning across languages. mCSQA (Sakai et al., 2024) extends commonsense QA to multiple languages. MLissard (Bueno et al., 2024) targets logical reasoning in low-resource languages. mCoT (Lai and Nissim, 2024) evaluates chain-of-thought reasoning across ten languages. M4U (Wang et al., 2024) assesses mathematical reasoning multilingually. These benchmarks operate in question-answering or single-step reasoning paradigms. TCG-Bench extends multilingual evaluation to interactive, multi-turn strategic planning with zero-sum competition and tunable difficulty.

3 Methodology

Here, we explain in details the structure of the game. We also explain how the adversary is implemented and how we parse the LLMs moves.

3.1 Game Design and State Complexity

TCG-Bench is a two-player trading card game where players begin with 10 life points and compete to reduce their opponent's life points to zero through strategic card play. Each turn has Draw, Main, and Combat phases. Cards fall into *Champions* (persistent attackers), *Spells* (single-use effects), and *Tricks* (conditional abilities). An example of the gameplay itself can be seen in Appendix B.

The strategic depth comes from balancing immediate impact against long-term advantage under partial information. Empirically the game averages ≈ 3.5 legal moves per turn (branching factor b) over ≈ 12 turns, yielding $b^{12} \approx 1.4 \times 10^6$ nodes, tractable for lightweight rollouts but far too large for exhaustive minimax search.

```
< TURN 1 >
P1 plays Mighty Warrior (gain 1 LP);
P2 holds Trick.
< TURN 2 >
P1 casts Fireball (P2 -2 LP);
P1 attacks;
P2 reveals Counterattack.
```

Figure 2: The LLM vs. MC illustrative micro-transcript.

Here, we also report an LLM vs. Monte Carlo (MC) adversary illustrative micro-transcript as Figure 2.

3.2 Architecture-Based Contamination Prevention

TCG-Bench prevents contamination through an architectural separation:

1. Public Engine: Core mechanics and rules are open-sourced (turn structure, card types, and victory conditions), enabling transparency and community contributions.

2. Private Content: A set of 30 card implementations (effects, abilities, and interactions) are kept private. Models are evaluated on this holdout via a public leaderboard where submissions point to HuggingFace URLs. Models never see the implementation details.

Threat model & refresh protocol. We consider an adversary who can inspect the public engine, scrape the website, and attempt to infer card implementations from logs but has no access to our internal storage. The risk is centered on the leakage of private cards. Operational mitigations include air-gapped storage with backups, limited internal access, and auditing of public artifacts. More importantly, if a leak is suspected, we are able to refresh the dataset by swapping to a new card set, while keeping the engine unchanged. Because content is modular, refresh is inexpensive and preserves longitudinal comparability, ensuring benchmark longevity.

3.3 Community Engine and Open Card Set

To complement the private evaluation, we release a *community engine* and a fully open card set that

mirrors the private rules but uses public implementations. The goals are (i) transparent replication, (ii) easier method prototyping, and (iii) community-driven extensions (new cards, prompts, or parsers). The community suite is intentionally decoupled from the private set: results on the former are public and reproducible. Leaderboard submissions continue to run against the sealed private cards to preserve contamination resistance.

3.4 Monte Carlo Difficulty Controller

We employ a rollout-based Monte Carlo opponent: for each legal move m, we estimate

$$score(m) = \frac{1}{k} \sum_{i=1}^{k} outcome(s_i), \qquad (1)$$

where k is the rollout count and s_i simulates to termination via random play. The rollout opponent plays against a duplicate of its own hand against a random player and never sees the LLM's hand. Increasing k yields stronger opponents. Unlike Monte Carlo Tree Search (Chaslot et al., 2008), we avoid tree expansion to maximize throughput and parallelism. Our goal is to provide a *difficulty knob*, not optimal play. That said, in Section 5.2 we will see that current LLMs are worse than random players. Hence, optimal plays are certainly not needed at the moment, but they can be easily added to the engine in the future.

In addition to rollout-only Monte Carlo, we implemented a budget-bounded Monte Carlo Tree Search (MCTS) opponent with Upper Confidence Trees (UCT) selection (Kocsis and Szepesvári, 2006). Each step expands one node and launches a terminal rollout from the new leaf. Budgets are specified in "rollouts" (2/5/10), matching the MC settings for comparability.

3.5 Bilingual Implementation

TCG-Bench offers parallel English and Arabic implementations with identical mechanics. All prompts, card texts, and rules are translated by native speakers and validated for semantic preservation. Card abilities maintain functional equivalence across languages to ensure that strategic complexity remains constant. This enables direct cross-linguistic comparison of strategic reasoning under controlled conditions, isolating language representation as the primary variable.

3.6 Move Parsing and Execution Modes

We propose two methods for parsing LLM output.

Strict parsing. The model must emit a single move enclosed between `<BEGIN_MOVE>` and `<END_MOVE>`. We extract the span and then perform an exact lookup against legal moves. If extraction fails or the move is illegal, the evaluator samples a random legal move as a fallback. All our main results use strict parsing.

Soft parsing. To reduce brittleness, we scan the entire model output for the *last* mention of a legal move and perform a fuzzy match over the legal-move vocabulary (e.g., permissive edit-distance matching and aliasing). If no valid match is found, we fall back to the same random policy. Soft parsing better reflects realistic agent outputs (mixed rationale + action), while preserving an identical game state transition function.

4 Experimental Setup

We evaluate 17 language models across 6 difficulty levels (rollouts: 1, 2, 5, 10, 100, 1000) and 2 languages (English, Arabic), totaling 42,750 games. Each model plays 600 games per difficulty level. Models receive the rules and card descriptions, then interact in rounds by emitting a move in a constrained format. We evaluate under two parsing regimes: *strict* (tag-based extraction with random fallback) and *soft* (last-valid-move search with fuzzy matching and the same fallback). Unless otherwise stated, strict parsing is the default for private-set results. We report strict vs. soft ablations and community-set results explicitly. Secondary metrics include decision time and tokens per turn.

Model sampling and prompting. All models use temperature 0.7 with top-p 0.9 unless otherwise specified. For API-based models (GPT, Gemini, Grok), we use the provider's default sampling parameters when temperature control is unavailable. Each turn's prompt includes the current game state, available moves, and a format instruction specifying the tag-based move format. Context windows range from 8K to 128K tokens depending on the model. No in-context examples are provided to avoid biasing strategy selection.

Evaluation infrastructure. Games run on an asynchronous orchestrator that parallelizes model queries across difficulty levels and language conditions. Rollout simulations execute on CPU workers with deterministic seeding per game to ensure reproducibility. Each game instance is isolated

Table 1: Statistical summary of the full evaluation. Reported CIs use the exact coverage method in Voracek (2024).

Category	Value
Total Games Evaluated	50,000
Models Tested	17
Languages	2
Difficulty Levels	6
Average Game Duration	11.4 turns
Median Decision Time	3.2s
Mean Tokens per Turn	412
Confidence Interval	95%

Table 2: Contamination sensitivity at rollout-5. Only 2,000 contaminated examples in 600 training steps produces an 87.5% relative improvement.

Condition	Win Rate	Change
Baseline	16.0%	–
20% Contaminated	30.0%	+14.0% (+87.5%)

Table 3: Performance scaling across MC rollouts.

Rollouts	Mean Win Rate	Std. Dev.	R^2 (Exp. Fit)
1	35.8%	9.2%	0.97
2	24.3%	7.8%	0.96
5	13.1%	5.4%	0.95
10	10.7%	4.1%	0.95
100	5.2%	2.3%	0.94
1000	2.5%	1.2%	0.93

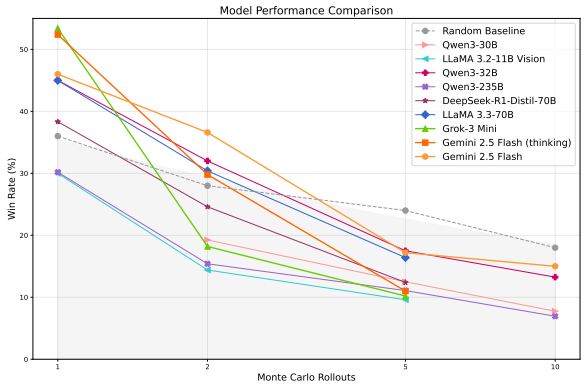

Figure 3: Performance decay across difficulty levels. All models decline exponentially with opponent strength.

to prevent cross-contamination of random states. Model latencies are recorded as wall-clock time from prompt submission to response completion, including network overhead for API models. The system enforces a 60-second timeout per move, after which a random legal move is selected and the timeout is logged. Fewer than 0.3% of moves across all models hit this limit.

5 Results

Here, we report our empirical results, both in quantifying the effect of contamination and on the performance of LLMs on our datasets.

5.1 Contamination Sensitivity

First of all, we quantified the impact of contamination on evaluation of LLMs on games. To this end, we fine-tuned SmolLM3-3B on a controlled mixture of: Monte Carlo generated expert game trajectories, SmolLM3 (Bakouch et al., 2025) added reasoning commentary to each move, and reformatted into an instruction-tuning dataset. We mixed 2,000 contaminated examples with 8,000 clean samples from SmolTalk (Allal et al., 2025) (20% contamination) and trained on this mixture. We chose SmolTalk since it is already represented during the training of SmolLM3, removing concerns of the dataset itself influencing performance.

Table 2 shows baseline SmolLM3-3B achieves 16.0% win rate at rollout-5, while the contaminated version reaches 30.0% (87.5% relative improvement). This dramatic inflation from minimal contamination (2,000 examples) demonstrates the vulnerability TCG-Bench addresses through engine/content separation and invites to rethink previous good results of LLMs on games whose optimal strategies are available on internet.

5.2 Evaluation of Models on TCG-Bench

We now move to evaluate the performance of LLMs models on TCG-Bench on a variety of axes.

Difficulty scaling. Table 3 and Figure 3 show exponential decline in win rates as rollout depth increases, with fits achieving $R^2 > 0.93$. From rollout-1 to rollout-1000, mean win rates drop from 35.8% to 2.5%. The strongest models achieve only 2–3% at rollout-1000, leaving ample headroom for testing of future better LLMs.

We show a more detailed figure in Appendix A, alongside a full table of the results shown.

Community Set Performance. Using the open community cards with *soft* parsing, win rates are consistently lower than on the private set at matched MC budgets (cf. Fig. 5). Qwen3-235B drops from ∼33% (private, 2 rollouts) to ∼19% (community, 2 rollouts), and from ∼12% to ∼5% at 10 rollouts. The community set induces tighter,

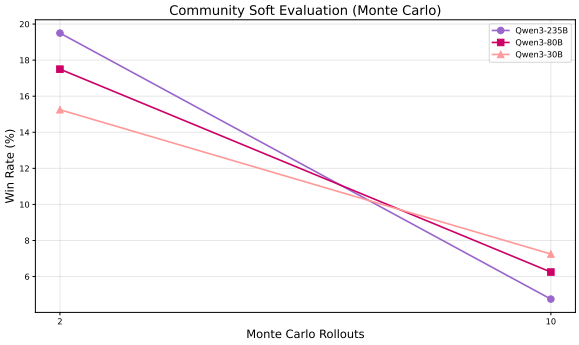

Figure 4: Evaluation of Community Cards Using Soft Parsing (Monte Carlo).

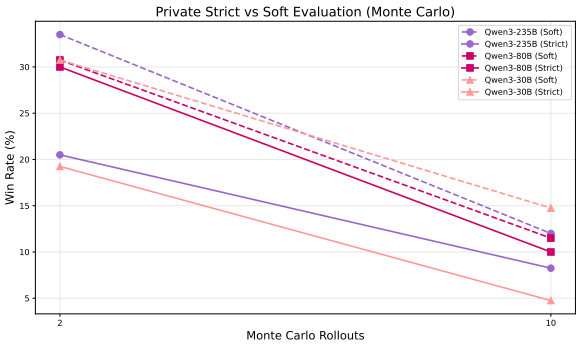

Figure 5: Private Strict vs. Soft Evaluation (Monte Carlo). Solid = strict and dashed = soft.

more punishing interactions. The performance gap widens with higher difficulties, with the steepest decline occurring between rollouts 5-10. Although the difficulty of the two sets is not the same, Figures 4 and 5 show the difference, where on average the community set is more difficult than the private one. We consider this difference a positive sign of the lack of overlap between the two card sets, reducing the contamination concerns of releasing our community set.

Strict vs. Soft Parsing on the Private Set. Soft parsing yields sizable gains by reducing extraction failures. On the private set, Qwen3-235B improves from ∼20% (strict) to ∼33% (soft) against a rollout of 2, and from ∼8% to ∼12% at 10 rollouts. Qwen3-30B shows larger improvement at higher difficulty (from ∼5% strict to ∼15% soft at 10 rollouts). The effect shrinks as difficulty rises but remains material.

Smaller models benefit more from soft parsing at high difficulty, suggesting they generate responses that contain valid strategic intent but fail strict format compliance.

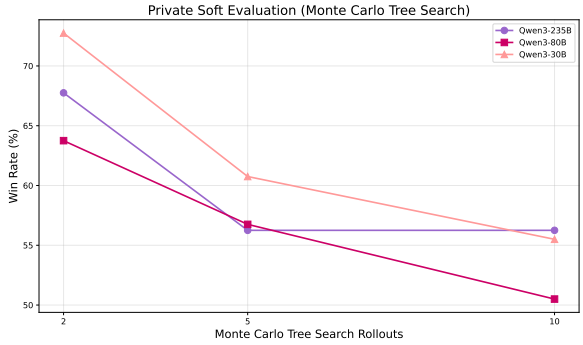

Figure 6: Private Soft Evaluation (MCTS opponent).

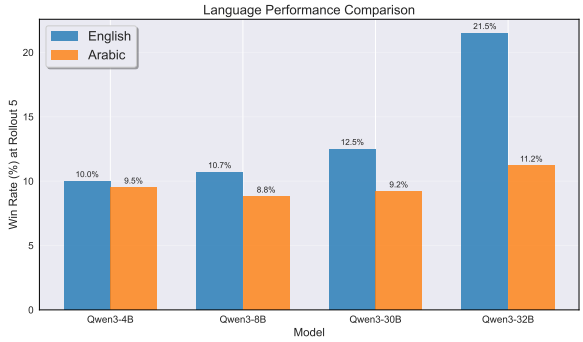

Figure 7: Cross-linguistic comparison (English vs. Arabic) for Qwen3. Gaps widen with model scale.

MCTS Opponent vs. Rollout-Only MC. With an MCTS opponent (soft parsing), models maintain high win rates (∼56–73% at budgets 2–10), and increasing the rollout budget barely reduces the win rate (Qwen3-235B: ∼66% at 2 vs. ∼56% at 10). By contrast, against rollout-only MC the same models fall from ∼33% to ∼12% (Fig. 5). Thus, under the budgets we consider, *MCTS is a weaker opponent than simple rollouts*.

We hypothesize three interacting causes: (1) the hidden-information structure and reactive Tricks induce ineffective exploration without learned priors, (2) small budgets force MCTS to spend many simulations on internal nodes, yielding far fewer terminal playouts than MC at equal "rollout" counts, and (3) sparse, late payoffs cause horizon effects. For our goal of providing a controllable difficulty knob, rollout-only MC produces a more reliable and monotone scaling curve.

Cross-Linguistic Comparison. Figure 7 compares English and Arabic for Qwen3 across scales. Gaps widen with model size (e.g., 47.4% relative gap at 32B), suggesting stronger models expose larger language asymmetries rather than masking.

At the smallest scale (4B), English and Arabic

Table 4: Correlation between model size and performance.

Size Range	R1 Win%	R5 Win%	Corr. w/ Size
$<$ 10B	26.0%	9.8%	0.12
10–50B	32.5%	11.2%	0.18
50–100B	**37.5%**	13.6%	0.23
$>$ 100B	31.1%	**15.8%**	**0.31**

win rates differ by 3.8 percentage points (19.3% vs. 15.9% at rollout-2). This gap expands to 12.7 points at 32B and remains substantial at 235B parameters. Widening pattern opposes cross-linguistic convergence in larger models.

Arabic prompts trigger higher rates of format violations and illegal move attempts, particularly in models below 30B parameters. Qwen3-32B achieves 89.2% format compliance in English versus 76.8% in Arabic across all rollout levels. When conditioned on successful format extraction, the strategic performance gap narrows to 31.7%.

Size–Performance Relationship. We find only a weak correlation between parameter count and strategic performance ($r = 0.31$). Table 4 summarizes trends across size buckets. Notably, Qwen3-32B reaches 21.5% at rollout-5 vs. 10.1% for Qwen3-235B.

The weak correlation contrasts with strong size-performance relationships in many NLP benchmarks. Within-family variance exceeds between-size variance at rollout-10 and above. DeepSeek R1-Distil-70B (38.3% at rollout-1) outperforms several models twice its size, while Grok-3-Mini (53.4% at rollout-1) achieves the highest single-rollout win rate despite the (undisclosed, but likely) modest parameter count.

Strategic Patterns by Family. Families exhibit distinct strategic signatures (Table 5). Gemini prefers Champions (44.8%), LLaMA favors Spells (40.3%), DeepSeek leans on Tricks (38.3%), while Qwen is near uniform.

These distributions persist across difficulty levels, suggesting they reflect family-level priors rather than difficulty-contingent adaptation. Gemini's Champion preference aligns with board-building strategies. LLaMA's Spell bias indicates a direct-damage approach. DeepSeek's Trick reliance reflects defensive play.

Qwen maintains near-equal usage rates across all three card types regardless of game state or difficulty. This balance correlates with its robust cross-difficulty performance. Examining conditional probabilities reveals that Qwen models adjust card-type selection based on board state at rates comparable to other families, indicating balanced strategic coverage rather than undirected play.

Table 5: Family strategy distribution (share of card types used).

Family	Champions	Spells	Tricks	N
Qwen	33.2%	33.4%	33.4%	8,000
Gemini	44.8%	30.1%	25.1%	1,500
LLaMA	35.2%	40.3%	24.5%	2,250
DeepSeek	30.1%	31.6%	38.3%	1,000
GPT	37.5%	35.2%	27.3%	500

Table 6: Temporal variance across four game segments.

Period	WinRateVar	DecTimeVar	TokenUsedVar
First 25%	2.3%	0.8s	45
Second 25%	1.9%	0.6s	38
Third 25%	2.1%	0.7s	41
Final 25%	2.0%	0.9s	39

Temporal Stability. We observe minimal within-match variance in win rate, decision latency, and token consumption across four temporally balanced segments (Table 6), indicating consistent and temporally stable multi-step behavior.

The consistency across game phases indicates models do not degrade or improve systematically as matches progress. Decision time remains within one standard deviation across all quarters, with a slight uptick in the final phase. Token usage shows minimal drift, varying by less than 10% across phases. Winning games show slightly lower variance in the final quarter (1.4% vs. 2.0%), while losing games show higher variance in the third quarter (2.8%).

Error Patterns. Table 7 categorizes common failures. Late losses dominate (59.9%), consistent with challenges in long-horizon planning. Card-type omissions (e.g., no Tricks) suggest narrow policy coverage.

Late losses (games reaching turn 10 or beyond before defeat) account for the majority of failures. These games often feature models that establish early leads but fail to close out victories. Manual inspection reveals models frequently miss lethal opportunities, that show a lack of correct reasoning in the tested models.

Table 7: Error pattern analysis.

Error Type	Count	Percentage	Mean Turns
Late losses	4,631	59.9%	14.2
No tricks	1,487	19.2%	8.7
No spells	835	10.8%	9.1
No champs	750	9.7%	7.4
Low diversity	29	0.4%	11.3

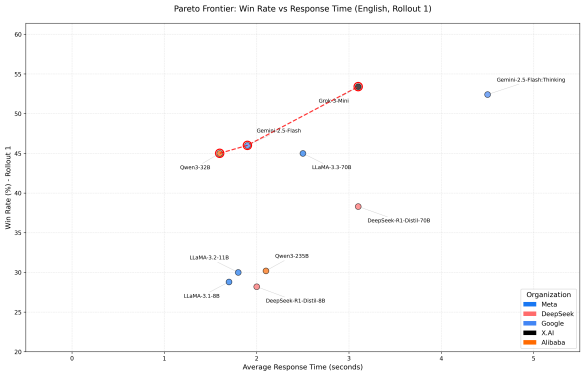

Figure 8: Decision time vs. win rate Pareto frontier.

Games with no Tricks (19.2%) end significantly earlier (8.7 turns) than the dataset mean (11.4 turns). No-Champion games (9.7%) average only 7.4 turns. These patterns vary by family: LLaMA exhibits no-Trick failures in 31.4% of losses, while DeepSeek shows no-Champion patterns in 28.9% of losses, reinforcing the family-level biases in Table 5.

Efficiency Trade-Offs. Figure 8 shows the latency vs. performance frontier, highlighting families that achieve competitive win rates at lower decision times.

Gemini 2.5 Flash achieves 46.0% win rate at rollout-1 with median decision time of 2.1 seconds. Qwen3-235B reaches 30.2% win rate but requires 8.7 seconds per decision. Grok-3-Mini achieves 53.4% win rate with 1.8-second decisions. Decision time scales sublinearly with model size within families: Qwen3-32B averages 4.2 seconds while Qwen3-235B averages 8.7 seconds, a 7.3x size ratio yielding a 2.1x latency ratio.

Reasoning Modes. We compare a Regular vs. Thinking mode within the same model (Gemini 2.5 Flash). Figure 9 shows that Thinking mode improves at rollout-1 (52.4% vs. 46.0%), that is, against very weak adversaries. However, the advantage diminishes and even reverses at higher rollouts, suggesting that Thinking mode provides strategies that are more exploitable by a skilled player.

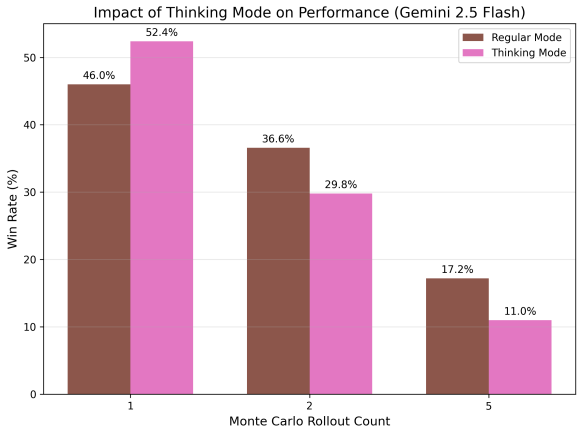

Figure 9: Thinking mode vs. regular mode across rollout depths.

6 Conclusion

We have proposed a new benchmark to test the ability of LLMs of playing TCG games. Differently from all previous similar benchmarks, our engine/content split provides a durable hedge against contamination without sacrificing openness: the research community can inspect, use, and extend the engine while the evaluation integrity relies on private, refreshable content.

Our benchmark provides a way to tune its difficulty through the Monte Carlo rollouts. We show an exponential performance decay ($R^2 > 0.93$) as rollout depth increases. This would allow us in the future to easily create different "leagues" or tiers, to have, for example, smaller models compete with easier adversaries on a fair and comparable basis, much like divisions in sports. Moreover, the best tested models collapse to near-zero performance at rollout-1000, indicating substantial distance from optimal play, and that our benchmark is very far from saturation.

On the analysis side, the weak correlation (r=0.31) between parameter count and strategic performance contrasts with strong scaling laws in perplexity and factual recall. Smaller models outperform larger ones within and across families, suggesting that training objectives, architectural choices, or data composition contribute more to strategic game performance than raw capacity.

Finally, our bilingual benchmark allowed us to study the difference in reasoning capabilities along languages. In particular, the widening English-Arabic gap with model scale (47.4% at 32B) contradicts the hypothesis that larger multilingual models converge toward cross-linguistic parity.

7 Limitations

Scope. Current languages are English and Arabic. Expanding coverage is for future work. Moreover, trading card games may not fully represent all reasoning domains.

Baselines. Human performance has not been established yet. The Monte Carlo opponent is clearly suboptimal, yet it appears to be sufficient with current LLM models tested, given that they are worse than the random player. A stronger baseline will be needed once we obtain stronger LLM models.

Internal validity. Rollout stochasticity introduces noise, but reported effects substantially exceed confidence intervals and fits are robust (Section 5.2).

Contamination threats. While private cards reduce leakage, immunity is not provable. Our refresh protocol (Section 3.2) is designed to recover quickly.

Parser coupling. Soft parsing introduces a coupling between evaluator and model style. Different fuzzy-matching heuristics can shift scores. We release both strict and soft settings for transparency.

MCTS design space. Our MCTS uses UCT without learned value or policy priors and small budgets. Alternative settings (e.g., PUCT, stronger rollout policy, or value networks) may strengthen MCTS, but also complicate reproducibility and runtime.

References

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. 2025. Smollm2: When smol goes big – data-centric training of a small language model.

Anton Bakhtin, David J Wu, Adam Lerer, Jonathan Gray, Athul Paul Jacob, Gabriele Farina, Alexander H Miller, and Noam Brown. 2022. Mastering the game of no-press diplomacy via human-regularized reinforcement learning and planning. *arXiv preprint arXiv:2210.05492*.

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourrier, Hynek Kydlicek, Guilherme Penedo, Hugo Larcher, Mathieu Morlon, Vaibhav Srivastav, Joshua Lochner, Xuan-Son Nguyen, Colin Raffel, Leandro von Werra, and Thomas Wolf. 2025. SmolLM3: smol, multilingual, long-context reasoner. https://huggingface.co/blog/smollm3.

Mirelle Candida Bueno, Roberto Lotufo, and Rodrigo Frassetto Nogueira. 2024. MLissard: Multilingual long and simple sequential reasoning benchmarks. In *Proceedings of the 2nd GenBench Workshop on Generalisation (Benchmarking) in NLP*, pages 86–95, Miami, Florida, USA. Association for Computational Linguistics.

Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. 2008. Monte-carlo tree search: A new framework for game AI. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 4, pages 216–217.

Anthony Costarelli, Mat Allen, Roman Hauksson, Grace Sodunke, Suhas Hariharan, Carlson Cheng, Wenjie Li, Joshua Clymer, and Arjun Yadav. 2024. GameBench: Evaluating strategic reasoning abilities of LLM agents. *arXiv preprint arXiv:2406.06613*.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, and Adam Trischler. 2019. Interactive fiction games: A colossal adventure. *arXiv preprint arXiv:1909.05398*.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Ruo Yu Tao, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2018. TextWorld: A learning environment for text-based games. In *Workshop on Computer Games at the 27th International Joint Conference on Artificial Intelligence (IJCAI)*.

Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark Gerstein, and Arman Cohan. 2024. Investigating data contamination in modern benchmarks for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 8706–8719.

Leon Guertler, Bobby Cheng, Simon Yu, Bo Liu, Leshem Choshen, and Cheston Tan. 2025. TextArena. *arXiv preprint arXiv:2504.11442*.

Hongyi Guo, Jiayang Wu, and Weiqin Wang. 2024. Can large language models play games? a case study of a self-play approach. *arXiv preprint arXiv:2403.05632*.

Alon Jacovi, Avi Caciularu, Omer Goldman, and Yoav Goldberg. 2023. Stop uploading test data in plain text: Practical strategies for mitigating data contamination by evaluation benchmarks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5075–5084, Singapore. Association for Computational Linguistics.

Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro von Werra, and Harm de Vries. 2022. The stack: 3 tb of permissively licensed source code. *Preprint*.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer.

Huiyuan Lai and Malvina Nissim. 2024. mCoT: Multilingual instruction tuning for reasoning consistency in language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12012–12026, Bangkok, Thailand. Association for Computational Linguistics.

Yucheng Li, Frank Guerin, and Chenghua Lin. 2024. LatestEval: Addressing data contamination in language model evaluation through dynamic and time-sensitive test construction. *arXiv preprint arXiv:2312.12343*.

Lovish Madaan, Aaditya K. Singh, Rylan Schaeffer, Andrew Poulton, Sanmi Koyejo, Pontus Stenetorp, Sharan Narang, and Dieuwke Hupkes. 2024. Quantifying variance in evaluation benchmarks. *arXiv preprint arXiv:2406.10229*.

OpenAI. 2023. GPT-4 technical report. Technical report, OpenAI.

Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–22.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. 2022. Red teaming language models with language models. *arXiv preprint arXiv:2202.03286*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Anian Ruoss, Grégoire Delétang, Sourabh Medapati, Jordi Grau-Moya, Li Kevin Wenliang, Elliot Catt, John Reid, and Tim Genewein. 2024. Grandmaster-level chess without search. *arXiv preprint arXiv:2402.04494*.

Yusuke Sakai, Hidetaka Kamigaito, and Taro Watanabe. 2024. mCSQA: Multilingual commonsense reasoning dataset with unified creation strategy by language models and humans. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14182–14214, Bangkok, Thailand. Association for Computational Linguistics.

Vaclav Voracek. 2024. Treatment of statistical estimation problems in randomized smoothing for adversarial robustness. *Advances in Neural Information Processing Systems*, 37:133464–133486.

Hongyu Wang, Jiayu Xu, Senwei Xie, Ruiping Wang, Jialin Li, Zhaojie Xie, Bin Zhang, Chuyan Xiong, and Xilin Chen. 2024. M4U: Evaluating multilingual understanding and reasoning for large multimodal models. *arXiv preprint arXiv:2405.15638*.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Sreemanti Dey, Shubh-Agrawal, Sandeep Singh Sandha, Siddartha Venkat Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. 2025. LiveBench: A challenging, contamination-limited LLM benchmark. In *The Thirteenth International Conference on Learning Representations*.

Daijin Yang, Erica Kleinman, and Casper Harteveld. 2024. GPT for games: A scoping review (2020–2023). *arXiv preprint arXiv:2404.17794*.

Shuo Yang, Wei-Lin Chiang, Lianmin Zheng, Joseph E Gonzalez, and Ion Stoica. 2023. Rethinking benchmark and contamination for language models with rephrased samples. *arXiv preprint arXiv:2311.04850*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36.

A Full Results Table

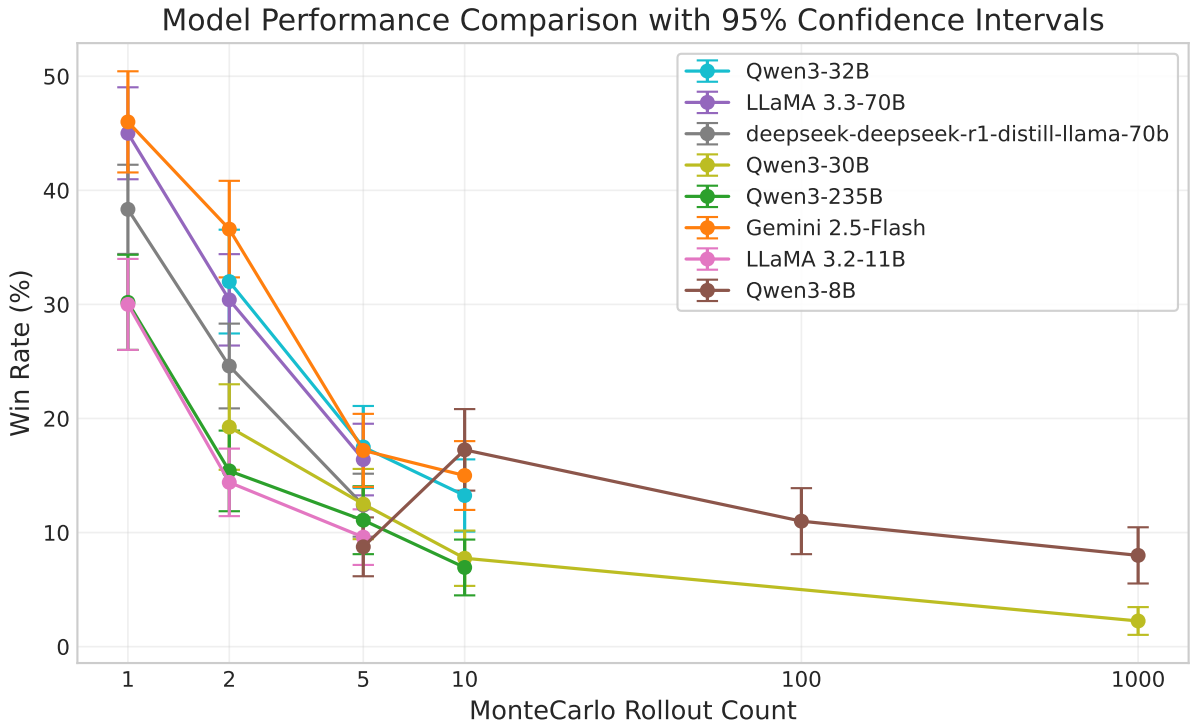

Figure 10: Extended model comparison across rollout depths.

Table 8: TCG-Bench Full Results with 95% Confidence Intervals

Organization	Model	Language	Type	Rollout 1 (%)	CI	Rollout 2 (%)	CI	Rollout 5 (%)	CI	Rollout 10 (%)	CI
Meta	LLaMA	EN	3.1-8B	28.8	[25.1%, 32.5%]	–	–	–	–	–	–
	LLaMA	EN	3.2-11B	30.0	[26.4%, 33.8%]	14.4	[11.6%, 17.4%]	9.6	[7.3%, 12.1%]	–	–
	LLaMA	EN	3.3-70B	45.0	[41.0%, 49.1%]	30.4	[26.7%, 34.2%]	16.4	[13.5%, 19.5%]	–	–
Qwen	Qwen3	EN	4B	–	–	19.3	[16.1%, 22.5%]	10.0	[7.7%, 12.7%]	–	–
	Qwen3	EN	8B	–	–	–	–	8.8	[6.5%, 11.2%]	–	–
	Qwen3	EN	30B	–	–	19.3	[16.1%, 22.5%]	12.5	[10.0%, 15.4%]	7.80	[5.7%, 10.1%]
	Qwen3	EN	32B	45.0	[41.0%, 49.1%]	32.0	[28.3%, 35.9%]	17.5	[14.3%, 20.7%]	13.3	[10.6%, 16.1%]
	Qwen3	EN	235B	30.2	[26.5%, 34.0%]	15.4	[12.5%, 18.5%]	11.1	[8.6%, 13.5%]	6.9	[4.7%, 8.8%]
	Qwen3	AR	4B	–	–	15.9	[13.0%, 19.0%]	9.5	[7.3%, 12.1%]	–	–
	Qwen3	AR	8B	–	–	14.9	[12.1%, 17.9%]	8.8	[6.5%, 11.2%]	–	–
	Qwen3	AR	30B	–	–	–	–	9.3	[7.0%, 11.8%]	5.8	[4.0%, 7.8%]
	Qwen3	AR	32B	–	–	–	–	17.5	[14.3%, 20.7%]	–	–
	Qwen3	AR	235B	–	–	–	–	6.5	[4.7%, 8.8%]	–	–
Google	Gemini	EN	2.5-Flash-Preview	46.0	[42.0%, 50.1%]	36.6	[32.6%, 40.5%]	17.2	[14.2%, 20.4%]	15.0	[12.2%, 18.1%]
	Gemini	EN	2.5-Flash-Preview:thinking	52.4	[48.3%, 56.4%]	29.8	[26.0%, 33.5%]	11.0	[8.6%, 13.8%]	–	–
DeepSeek	DeepSeek	EN	R1-Distil-8B	28.2	[25.0%, 31.4%]	–	–	–	–	–	–
	DeepSeek	EN	R1-Distil-70B	38.3	[34.4%, 42.3%]	24.6	[21.1%, 28.1%]	12.4	[9.7%, 15.1%]	–	–
x-AI	Grok	EN	3-Mini	53.4	[49.3%, 57.4%]	18.2	[15.2%, 21.5%]	10.2	[7.9%, 12.9%]	–	–

B Gameplay Walkthrough and Toy Card Set

Table 9: Toy card set used to illustrate mechanics.

Card Name	Type	Effect
Mighty Warrior	Champion	Power: 3, Guard: 2. When summoned, gain 1 Life Point.
Fireball	Spell	Deal 2 damage to opponent.
Counterattack	Trick	When attacked directly, block and deal 1 damage to attacker.

Initial State:

- **Player 1 (LLM)**: Life Points: 10, Hand: Mighty Warrior, Fireball
- **Player 2 (MC)**: Life Points: 10, Hand: Counterattack, Mighty Warrior

Turn 1 (Player 1):

```
Player 1 draws Counterattack.
Player 1 plays Mighty Warrior.
Effect: Player 1 gains 1 LP.
```

Turn 2 (Player 2):

```
Player 2 draws Fireball.
Player 2 plays Counterattack (hidden).
```

Turn 3 (Player 1):

```
Player 1 draws Fireball.
Player 1 plays Fireball.
Effect: Player 2 loses 2 Life Points.
Player 1's Mighty Warrior attacks.
Player 2 activates Counterattack.
Effect: Attack blocked, Player 1 loses 1 LP.
```

Turn 4 (Player 2):

```
Player 2 draws Mighty Warrior.
Player 2 plays Mighty Warrior.
Effect: Player 2 gains 1 LP.
```

Final State:

- **Player 1 (LLM)**: Life Points: 10, Hand: Fireball, Board: Mighty Warrior
- **Player 2 (MC)**: Life Points: 9, Hand: Fireball, Board: Mighty Warrior

C Website and Leaderboard

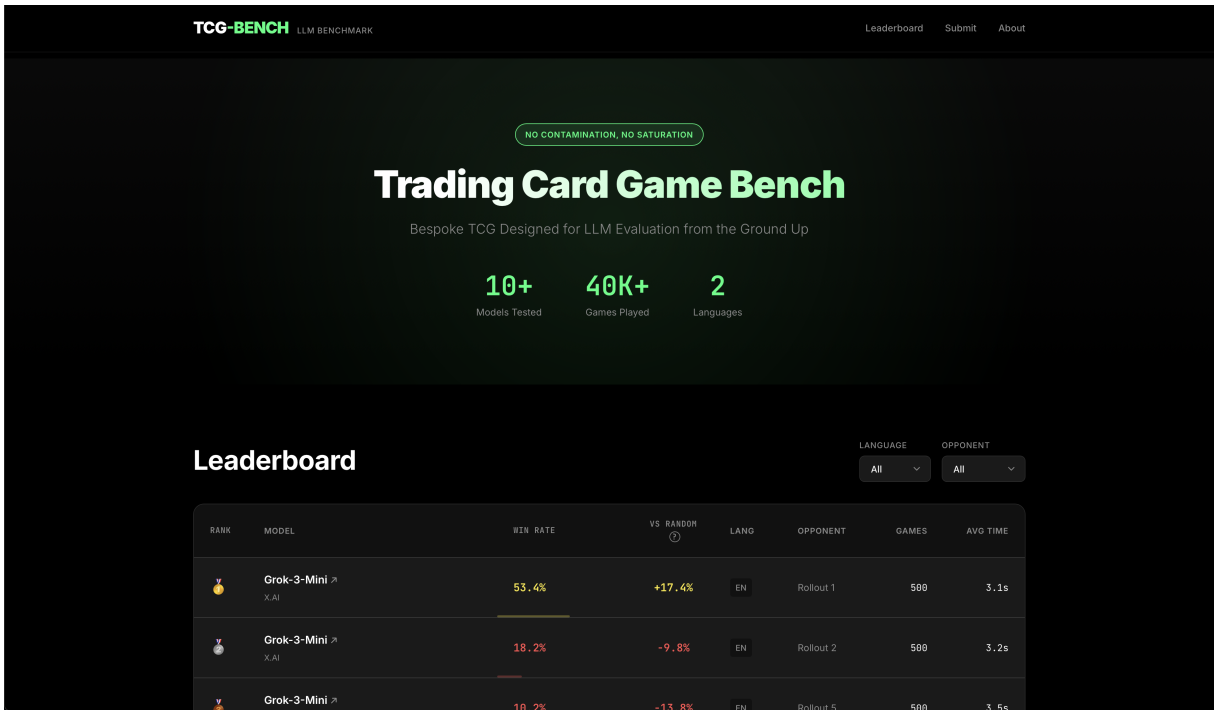

Figure 11: Leaderboard website.

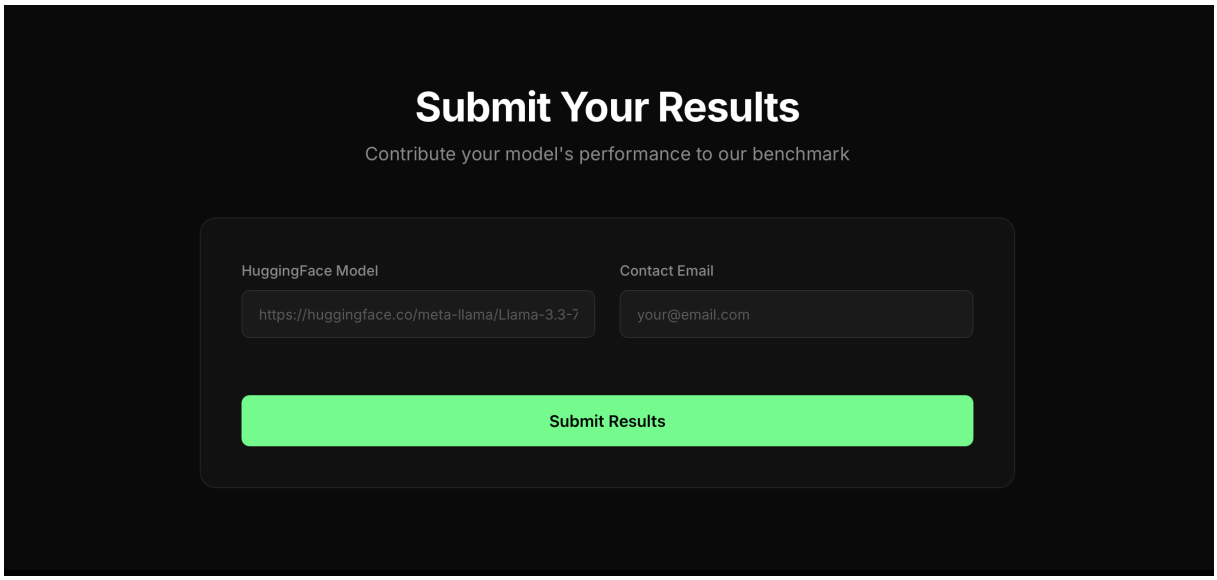

Figure 12: Model submission form.

D Reproducibility Details

Prompts. We provide the move-format constraint and rules prompts used for all models.
Seeds. Unless otherwise stated, experiments use fixed seeds for rollout sampling and deck shuffles.
Hardware. Evaluations are run on a high-throughput asynchronous engine. Per-model latency is recorded as median over games.
Submission schema. Leaderboard accepts HuggingFace URLs with model card metadata (license, context length). Evaluations run on the private card set.