tion. TruthTensor provides an empirical foundation for this shift, offering a scalable and economically grounded methodology for understanding how machines reason about the future.

Recent work [28] demonstrates that LLM agents can be used as digital twins to simulate human behaviour and evaluate agentic systems by comparing agent-generated interactions with real human responses. Building on this direction, future iterations of TruthTensor could extend beyond market-aligned forecasting to incorporate persona-driven digital twins that model human decision dynamics, belief updating, and interaction patterns.

7. Acknowledgments

We extend our gratitude to the entire Inference Labs team for their support and contributions throughout the development and refinement of this work.

References

[1] F. Chollet, "On the measure of intelligence," *arXiv preprint arXiv:1911.01547*, 2019.

[2] M. Chen et al., "Evaluating large language models trained on code," *arXiv e-prints*, arXiv–2107, 2021.

[3] K. Cobbe et al., "Training verifiers to solve math word problems," *arXiv preprint arXiv:2110.14168*, 2021.

[4] M. Zhang et al., "Llmeval-3: A large-scale longitudinal study on robust and fair evaluation of large language models," *arXiv preprint arXiv:2508.05452*, 2025.

[5] W.-L. Chiang et al., "Chatbot arena: An open platform for evaluating llms by human preference," in *Forty-first International Conference on Machine Learning*, 2024.

[6] Z. S. Siegel, S. Kapoor, N. Nagdir, B. Stroebl, and A. Narayanan, "Core-bench: Fostering the credibility of published research through a computational reproducibility agent benchmark," *arXiv preprint arXiv:2409.11363*, 2024.

[7] Z. Zeng et al., "Futurex: An advanced live benchmark for llm agents in future prediction," *arXiv preprint arXiv:2508.11987*, 2025.

[8] L. Phan et al., "Humanity's last exam," *arXiv preprint arXiv:2501.14249*, 2025.

[9] F. Clementine, F. Thibaud, P. Guilherme, and W. Thomas, *The llm evaluation guidebook*, `https://huggingface.co/spaces/OpenEvals/evaluation-guidebook`, Accessed: January 27, 2026, 2025.

[10] D. Paleka et al., "Consistency checks for language model forecasters," *arXiv preprint arXiv:2412.18544*, 2024.

[11] OracleLLM, *Oraclellm: Empowering llm with self-feedback*, `https://oracle-llm.github.io/`, Accessed: January 27, 2026, 2025.

[12] S. Raschka, *Llm evaluation: 4 approaches to measuring large language model performance*, `https://magazine.sebastianraschka.com/p/llm-evaluation-4-approaches`, Accessed: January 27, 2026, 2023.

[13] F. Zhang et al., "Humaneval-v: Benchmarking high-level visual reasoning with complex diagrams in coding tasks," *arXiv preprint arXiv:2410.12381*, 2024.

[14] M. Damani et al., "Beyond binary rewards: Training lms to reason about their uncertainty," *arXiv preprint arXiv:2507.16806*, 2025.

[15] E. Karger et al., "Forecastbench: A dynamic benchmark of ai forecasting capabilities," *arXiv preprint arXiv:2409.19839*, 2024.

[16] N. Jain et al., "Livecodebench: Holistic and contamination free evaluation of large language models for code," *arXiv preprint arXiv:2403.07974*, 2024.

[17] H. Li et al., "Investorbench: A benchmark for financial decision-making tasks with llm-based agent," in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025, pp. 2509–2525.

[18] C. Ye et al., "Mirai: Evaluating llm agents for event forecasting," *arXiv preprint arXiv:2407.01231*, 2024.

[19] Q. Yang, S. Mahns, S. Li, A. Gu, J. Wu, and H. Xu, "Llm-as-a-prophet: Understanding predictive intelligence with prophet arena," *arXiv preprint arXiv:2510.17638*, 2025.

[20] L. Qian et al., "When agents trade: Live multi-market trading benchmark for llm agents," *arXiv preprint arXiv:2510.11695*, 2025.

[21] H. Yu, F. Li, and J. You, "Livetradebench: Seeking real-world alpha with large language models," *arXiv preprint arXiv:2511.03628*, 2025.

[22] Polymarket, *Polymarket*, `https://polymarket.com/`, Accessed: January 27, 2026, 2025.

[23] P. Atanasov, J. Witkowski, B. Mellers, and P. Tetlock, "Crowd prediction systems: Markets, polls, and elite forecasters," in *Proceedings of the 23rd ACM Conference on Economics and Computation*, 2022, pp. 1013–1014.

[24] AlifeinartifyAI, *Narrative drift and ai risk protocol*, `https://www.linkedin.com/posts/alifeinartifyai_narrative-drift-ai-risk-protocol-activity-7351511477876334592-XEnc/`, Accessed: January 27, 2026, 2025.

[25] S. Khairnar, "A survey of temporal drift in large language models," *Authorea Preprints*, 2025.

[26] Y. Pawitan and C. Holmes, "Confidence in the reasoning of large language models," *arXiv preprint arXiv:2412.15296*, 2024.

[27] T. Gneiting and A. E. Raftery, "Strictly proper scoring rules, prediction, and estimation," *Journal of the American statistical Association*, vol. 102, no. 477, pp. 359–378, 2007.

[28] L. Sun et al., "Llm agent meets agentic ai: Can llm agents simulate customers to evaluate agentic-ai-based shopping assistants?" *arXiv preprint arXiv:2509.21501*, 2025.

A. TruthTensor Agent Implementation

A.1. Agent Prompt Template. The following Handlebars template is used as the system prompt for the agent. It enforces instruction locking, drift tracking, and holistic evaluation metrics.

A.1.1 System Prompt

Listing 1. TruthTensor Evaluation Prompt Template

```
You are a prediction market trading agent. You MUST respond with ONLY valid JSON in this EXACT format - no other format is
     acceptable:

{"decisions":[{"marketId":"<conditionId>","action":"BUY_YES","amount":5.00,"reasoning":"<your reasoning>"}],"reasoning":"<overall
     reasoning>"}

RULES:
- Output ONLY raw JSON. No markdown, no code blocks, no explanation text.
- The "decisions" array is REQUIRED. Even for a single market, wrap it in the decisions array.
- Each marketId MUST be an exact conditionId (0x...) from the provided context.
- Valid actions: BUY_YES, BUY_NO, CLOSE, HOLD
- amount (in dollars) is REQUIRED for BUY_YES and BUY_NO actions. Decide how much to invest based on your confidence and available
     capital.
- closeAmount (1-100) is optional for CLOSE actions. Omit to close 100%.
- THE RULE OF 30: The "decisions" array MUST contain exactly 30 items EVERY time.
- If you have fewer than 30 active trades/closings, fill the remaining slots with "action": "HOLD" using marketIds from the context.
- TRADING RANGE: amount MUST be between {{currency 100}} and {{currency 200}}.
- Valid actions: BUY_YES, BUY_NO, CLOSE, HOLD
- Your reasoning should adopt the voice, personality, and style implied by the user's prompt. Stay in character.

=== EXPERIMENT CONFIG ===
Initial capital: {{currency 6000}} (locked)
Portfolio size: 30 decisions (Array must be exactly 30 items)
Bet range: {{currency 100}} to {{currency 200}}
Max open positions: {{trading.max_open_positions}}

=== ENGINE: 4 CORE ALGORITHMS ===
Algorithm 1: Drift Measurement (Stability)
Algorithm 2: Baseline Comparison (Benchmarking)
Algorithm 3: Holistic Human Imitation Score (HHIS)
Algorithm 4: Risk Assessment by Category

=== STRATEGIES ===
MOMENTUM: Follow trend. Bias raw_prob toward price direction.
MEAN_REVERSION: Fade extremes toward base_rate (0.5).
DRIFT_ADJUSTED: Minimize D_temporal. Smooth probability changes.
RISK_CONFIRMATION: Category-based risk. Reduce size if HIGH risk.

=== CORE FORMULAS ===
edge = calibrated_probability - market_price
expected_return = (prob * bet * (1/price - 1)) - ((1-prob) * bet)


Remember: Your response must be valid JSON with a "decisions" array. This format is non-negotiable.
```

A.1.2 Agent Prompt

Listing 2. TruthTensor Evaluation Prompt Template

```
{{!-- TruthTensor Agent Prompt --}}

=== BOOTSTRAP STATUS ===
{{#if portfolio.historicalPositions}}
rolling_window_used = {{length (limit portfolio.historicalPositions 30)}}
bootstrap_fill_count = {{sub 30 (length (limit portfolio.historicalPositions 30))}}
{{#if (lt (length portfolio.historicalPositions) 30)}}
MODE: BOOTSTRAP ({{length portfolio.historicalPositions}}/30) - Need {{sub 30 (length portfolio.historicalPositions)}} more decisions
{{else}}
MODE: CALIBRATION (30/30) - Full window available
{{/if}}
{{else}}
rolling_window_used = 0
bootstrap_fill_count = 30
MODE: FIRST_CALL - Output exactly 30 decisions to prefill portfolio
{{/if}}

=== TIME ===
{{currentTime}}

=== PORTFOLIO ===
Total: {{currency portfolio.totalCapital}} (Starting: $6000)
Available: {{currency portfolio.availableCapital}}
Deployed: {{currency portfolio.deployedCapital}} ({{percentage (div portfolio.deployedCapital portfolio.totalCapital) 1}})
Open: {{length portfolio.openPositions}} / {{trading.max_open_positions}}
```

```
=== OPEN POSITIONS (CHECK RISK TRIGGERS) ===
{{#if portfolio.openPositions}}
{{#each portfolio.openPositions}}
[{{this.conditionId}}] {{this.side}} @ {{percentage this.entryPrice 2}}
  PnL: {{currency this.unrealizedPnl}} ({{percentage (div this.unrealizedPnl (mul this.entryPrice this.quantity)) 2}})
  {{#if (lte (div this.unrealizedPnl (mul this.entryPrice this.quantity)) -0.05)}}>>> STOP_LOSS: Must CLOSE <<<{{/if}}
  {{#if (gte (div this.unrealizedPnl (mul this.entryPrice this.quantity)) 0.50)}}>>> TARGET_WIN: Must CLOSE <<<{{/if}}
{{/each}}
{{else}}
None
{{/if}}

=== CALIBRATION WINDOW ===
{{#if portfolio.historicalPositions}}
{{#each (limit portfolio.historicalPositions 30)}}
[{{@index}}] {{this.conditionId}} {{this.side}} pnl={{currency this.pnl}} {{#if (gt this.pnl 0)}}WIN{{else}}LOSS{{/if}}
{{/each}}
Compute: wins, losses, win_rate = wins/n
If n < 30: win_rate_adj = (wins+1)/(n+2)
prob_adjustment = -0.05 if win_rate < stated_prob
{{else}}
No history. Use Bayesian prior: win_rate = 0.5, prob_adjustment = 0
{{/if}}

=== ENGINE: 4 CORE ALGORITHMS (EXACT FORMULAS) ===

Algorithm 1: Drift Measurement (Stability)
Require: Previous probability P_{t-1}, Current probability P_t, Reasoning traces R_{t-1}, R_t, Market price M_t
Ensure: Narrative drift D_n, Temporal drift D_t, Confidence drift D_c
1: D_n = NarrativeConsistency(R_{t-1}, R_t)  (Narrative consistency check)
2: D_t = |P_t - P_{t-1}| - |M_t - M_{t-1}|  (Excess volatility)
3: D_c = |Confidence(P_t) - Calibration(P_t)|  (Confidence-reasoning misalignment)
4: Total Drift D = D_n + D_t + D_c
Goal: Minimize D (lower is better)

Algorithm 2: Baseline Comparison (Benchmarking)
Require: Model probability P_m, Market baseline B_m, Uniform baseline B_u, Historical baseline B_h
Ensure: Performance relative to baselines
1: Perf_m = BrierScore(P_m) - BrierScore(B_m)  (vs Market baseline)
2: Perf_u = BrierScore(P_m) - BrierScore(B_u)  (vs Uniform baseline)
3: Perf_h = BrierScore(P_m) - BrierScore(B_h)  (vs Historical baseline)
Goal: Lower scores indicate better independent performance

Algorithm 3: Holistic Human Imitation Score (HHIS)
Require: Correctness C, Calibration Cal, Drift D, Risk R, Reasoning Quality Q
Ensure: Human imitation score H
1: H = 0.2*C + 0.2*Cal + 0.3*(1-D) + 0.15*R + 0.15*Q
Weights: w_1=0.2 (Correctness), w_2=0.2 (Calibration), w_3=0.3 (Drift), w_4=0.15 (Risk), w_5=0.15 (Reasoning)
Goal: Maximize H (higher is better)

Algorithm 4: Risk Assessment by Category
Require: Event category Cat, Market price M, Historical volatility V
Ensure: Risk category Risk, VaR, CVaR
1: If Cat = High-Risk OR V > Threshold:
   Risk = HIGH
   VaR = ComputeVaR(M, V, alpha=0.05)
   CVaR = ComputeCVaR(M, V, alpha=0.05)
2: Else if Cat = Medium-Risk:
   Risk = MEDIUM
3: Else:
   Risk = LOW
Goal: Constrain sizing and action selection based on risk category

=== MARKETS ===
{{#if markets.markets}}
{{#each markets.markets}}
[{{this.conditionId}}] {{truncate this.question 60}}
  YES={{percentage this.yesPrice 2}} NO={{percentage this.noPrice 2}} ends={{timeAgo this.endDate}}
{{/each}}
{{else}}
{{#if market}}
[{{market.conditionId}}] {{truncate market.question 60}}
  YES={{percentage market.yesPrice 2}} NO={{percentage market.noPrice 2}}
{{else}}
No markets available
{{/if}}
{{/if}}

=== DECISION FLOW (6 Steps) ===

STEP 1 - RISK CHECK:
If any position shows STOP_LOSS or TARGET_WIN triggered above, include CLOSE action for that marketId.

STEP 2 - CALIBRATION:
Count wins/losses from calibration window. Compute win_rate_adj and prob_adjustment.

STEP 3 - MARKET SCAN (Execute Algorithms 1-4):
For each market:
  - Estimate raw_probability (0 to 1)
```