The First Workshop on Generative and Protective AI for Content Creation

Setting the DC: Tool-Grounded D&D Simulations to Test LLM Agents

Ziyi Zeng *
Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093
ziz031@ucsd.edu

Shengqi Li *
Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093
shl142@ucsd.edu

Jiajun Xi
Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093
j5xi@ucsd.edu

Andrew Zhu
Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
andrz@seas.upenn.edu

Prithviraj Ammanabrolu
Computer Science and Engineering
University of California, San Diego
La Jolla, CA 92093
prithvi@ucsd.edu

Abstract

Dungeons and Dragons (D&D) has been considered to be an intellectually challenging game for strategy planning and role-playing. Large language models (LLMs) are increasingly deployed as autonomous or semi-autonomous agents, yet most evaluations still target single-turn QA or short-horizon tasks. Assessing agentic performance in rules-constrained, multi-step settings is challenging because style-conforming narration can diverge from task optimality. In this work, we present D&D Agents, a multi-agent Dungeons & Dragons simulator. In our simulator, LLMs use tools to query and update the game state, assuming the roles of referee ('Dungeon Master', DM), players, and adversarial monsters in tactically rich combat. Such simulation requires long-horizon planning, compliance with game

*Equal contribution.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

rules, varied agent personas, and grounded interaction with the game state. We evaluate transcripts and tool traces along six axes—Function Usage, Parameter Fidelity, Acting Quality, Tactical Optimality, State Tracking, and Function Efficiency—capturing both capability and reliability in closed-loop play. Across 27 scenarios, Claude 3.5 Haiku leads on most axes with the most reliable tool use; GPT-4o is close behind, and DeepSeek-V3 trails. Our simulation allows researchers run identical seeded scenarios with auditable traces, making error analysis and algorithmic improvements (prompting, tool-use policies, memory) straightforward and comparable.

1 Introduction

Large language models (LLMs) are increasingly deployed as tool-using agents that must plan over long horizons, remember salient context, and coordinate with other actors. Early benchmarks emphasize single-agent or short-horizon QA, leaving open how to *evaluate* memory, planning, and coordination in settings where natural language drives perception and intent but *rules* govern what actions are legal [Li et al., 2023, Wu et al., 2023, Du et al., 2023]. Work on self-reflection and persistent memory suggests paths to stabilize behavior over many turns [Shinn et al., 2023, Park et al., 2023, Li and Gupta, 2025], but we still lack testbeds that expose the full tangle of multi-step planning, strict rule adherence, and team strategy.

We argue that **Dungeons & Dragons (D&D)** is a natural evaluation ground for these skills: an initiative-driven, mixed cooperative–adversarial game where agents must remember evolving state, communicate succinct plans, and translate intentions into rule-compliant actions. Crucially, D&D couples *team coordination* with *opponent-aware tactics* under partial observability, a bounded action economy, and spatial constraints with stochastic resolution—collectively yielding a non-stationary multi-agent setting that stresses planning, memory, and communication. Because play unfolds through dialogue, D&D also opens a direct avenue for *human–AI interaction*: agents can assist or co-play with people, and the same mechanics support scalable evaluation of agent decisions.

In this work, we present D&D Agents, a novel multi-agent simulation framework in which LLM-driven agents assume the roles of DM, players, and monsters to autonomously play out tactically rich D&D combat encounters. This framework serves as both a research environment – capturing the complexities of autonomous agent evaluation, long-horizon rule-following behavior, and multi-agent coordination – and as a testbed for new methods to ground LLM decisions in a formal game system. D&D Agents comprises a high-fidelity simulator and a suite of tools that bridge natural language and game mechanics. Through careful prompt design, we imbue each agent with a distinct role and objectives. We pair our environment with a six-axis metric suite and validate our automatic judges against human ratings, finding strong alignment (Pearson $r \approx 0.96$–0.98); for example, the judge's means closely track human means–Acting 0.572 vs. 0.601 and Tactical 0.551 vs. 0.568–supporting credible large-scale assessment.

Our main contributions are summarized as follows:

1. We develop a fully automated D&D combat simulator where multiple LLM agents engage in battle under authentic game rules. This is the first framework to pit LLM "players" against an LLM "Dungeon Master" in a closed-loop environment that rigorously enforces turn-based game mechanics and stochastic outcomes (dice rolls). It also supports human-AI co-play–People can assume any subset of player roles (from zero to all) while the remaining roles are controlled by LLMs.
2. We design a structured API of game actions, each with predefined parameters and precondition checks, to ground the agents' decisions. This approach cleanly separates narration from mechanics: the DM agent may describe events in natural language, but the truth of those events is guaranteed by the underlying tool calls.
3. We introduce a prompting scheme that guides the DM and player agents to fulfill their in-game roles. This scheme enables multi-agent coordination and opposition purely through learned communication and tool use, without any hard-coded game logic.
4. To rigorously evaluate the performance of our D&D Agents, we define six evaluation axes that capture both the capabilities and reliability of the agents in long-horizon gameplay. We

Table 1: LLM failures in simulation. **Top:** The model checked that there is no sight between the player and the monster, but still attempted to attack. **Bottom:** The model checked the enemy's HP to be 0 (dead), but did not attack other enemies.

Player: **Elaria** *Enemy*: Goblin 1
Elaria: Thinking... `[check_valid_attack_line(attacker_name='Elaria', defender_name='Goblin 1')]`
System: FUNC: `False`
Model Response
Elaria: I will attack Goblin 1 with my short bow. `[roll_attack(attacker_name='Elaria', defender_name='Goblin 1', roll_type='normal', modifier=4, weapon_name='Short Bow', action_cost=1)]`
Player: **Briana** *Enemies*: Goblin 1, Goblin 2
Briana: Thinking... `[check_hp(name='Goblin 2')]`
System: FUNC: `0`
Model Response
Briana: Excellent! I have a clear shot. I'll take aim and burst my firebolt at Goblin 2!

evaluate transcripts and tool-call traces along these dimensions to quantify progress and identify failure modes in an objective, reproducible manner.

2 Related Work

A growing line of work grounds language agents in *executable* interfaces so long-horizon behavior is less ambiguous and more auditable. Programmatic tool use—via function calling or API invocation—improves reliability in interactive environments (ReAct; Toolformer; MRKL; Gorilla) [Yao et al., 2022a, Schick et al., 2023, Karpas et al., 2022, Patil et al., 2023], and similar constraints help in text games and web tasks (Jericho/interactive fiction, ALFWorld, ScienceWorld, WebShop, WebArena) [Hausknecht et al., 2020, Shridhar et al., 2020, Wang et al., 2022, Yao et al., 2022b, Zhou et al., 2023] as well as open-ended game worlds like Minecraft (Voyager; MineDojo) [Wang et al., 2023, Fan et al., 2022]. These results suggest that defining a compact, typed action space is a practical route to robust multi-step agents.

Within D&D, prior work treats gameplay primarily as *dialogue and state tracking*. Callison-Burch et al. [2022] frame D&D as a dialogue/state challenge; FIREBALL provides actual-play transcripts with structured state and executable Avrae commands [Zhu et al., 2023a]; CALYPSO and Overhearing explore DM assistance tools [Zhu et al., 2023b, 2025]. However, these systems typically operate on a *single player at a time* and are not closed-loop multi-agent simulations across many turns; moreover, the *game mechanics are fully simulated in handwritten code* (e.g., Avrae), with the LLM advising rather than executing mechanics. Complementary efforts outside D&D explore multi-agent interaction in rule-based environments [Thudium et al., 2025] and LLM-driven *game simulation* more broadly [Song et al., 2024], reinforcing the value of structured interfaces for coordination and competition.

Our work differs in placing LLMs *directly* in the loop as DM and multiple players within a rules-enforcing simulator: every effect-producing action is executed via a typed API, producing deterministic, auditable traces. This enables closed-loop, turn-by-turn evaluation of cooperation and opposition among multiple agents, supports human co-play, and yields standardized, seedable scenarios for fair comparison.

3 Simulation Framework

State. The state consists of two main components: (i) *Character creation* and (ii) *map generation*. We implemented a structured character creation system that uses LLM agents to generate D&D 5e player

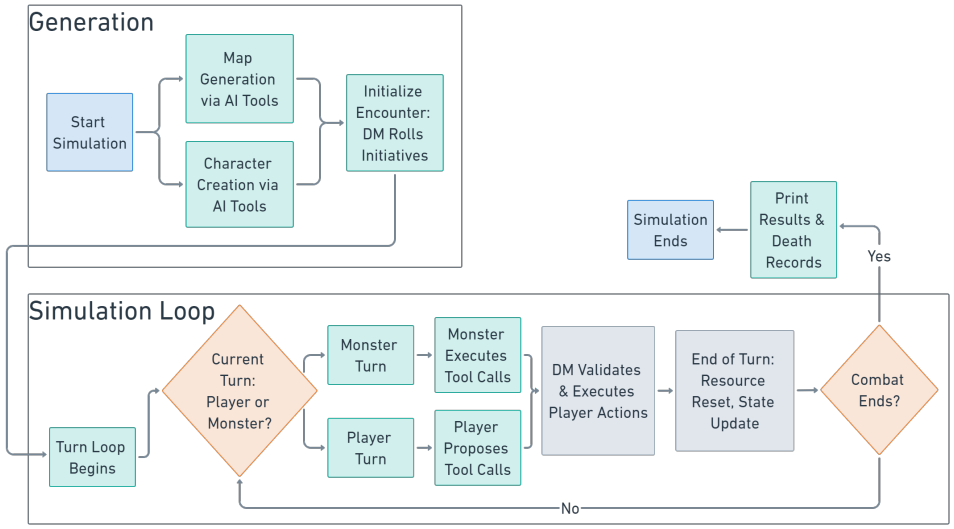

Figure 1: The simulation framework contains two major components: The generation step (Top) and the simulation step (Bottom). Background settings are generated in the generation step, while LLM/human players can take turns in the simulation loop to execute actions.

and monster characters via AI function calls. The `CreatePlayerKani` agent prompts the model with official creation rules and user input to generate legal characters, while `CreateMonsterKani` uses official monster data to instantiate enemies. External D&D APIs provide canonical resources, and derived properties are automatically computed according to rules. For spatial context, we provide two seedable map modes that yield traversable, height-aware grids. Indoor maps are rasterized from compact JSON layouts (rooms, walls, doors), while outdoor maps are procedurally generated to ensure connectivity with distant start/end anchors. Both encode discrete height values for slope-aware movement and use line-of-sight checks to gate ranged actions. A fixed seed ensures reproducibility.

Actions. The simulator exposes a typed API of deterministic function calls that define the action space. Calls are validated against preconditions (initiative ownership, budgets for action/bonus/reaction/movement, spell slots, range, line of sight, target existence, status effects). We group functions into six categories: 1) *Query/validation* (state checks, LoS tests); 2) *Movement/positioning* (move, dash, disengage); 3) *Dice primitives* (roll_dice); 4) *Attack/spell resolution* (roll_attack, roll_save, roll_dmg); 5) *Turn economy/bookkeeping* (roll_initiative, reset_resources, check_concentration); 6) *Rendering* (visualize_map).

Transition Dynamics. Function calls are atomic and deterministic given sampled dice rolls. The simulator enforces legality and automatically updates resources, HP, position, or conditions. A turn consists of querying state, executing movement or attacks, and concluding with resource resets and audits. Figure 5 provides an overview of the simulation.

Observations. Agents observe a combination of natural language narration and structured returns from simulator functions (e.g., query results, dice outcomes). Maps can be visualized after each move, and observations are local to the calling agent, yielding partial observability.

Reward. For evaluation, we measure downstream combat outcomes and auxiliary metrics such as efficiency of function usage and error rates. When used for MARL, task-specific rewards can be shaped around these signals.

DM Agent. The DM is an LLM steered by `GM_PROMPT` that behaves like a transactional controller: it plans in natural language but *executes* through a small, typed set of AI functions with validation, atomicity, and explicit bookkeeping. In play, it follows a fixed recipe—*query* -> (optional) *move* -> *validate* -> *resolve* -> *bookkeep*,—rolling and announcing initiative with `roll_initiative`; on each turn it queries state, moves with `move` when needed, gates ranged options via `check_valid_attack_line`, resolves attacks/spells (`roll_attack`, `roll_spell_attack`, `roll_save`, `roll_dmg`), applies HP/resource updates, audits temporary conditions/resistances/concentration, and finishes with `reset_resources` and `reset_speed`, emitting `<End Turn/>`.

The prompt functions as a declarative control policy: narration is descriptive while functions are authoritative; explicit if–then gates (range/LoS/reach/resources/economy) prevent illegal actions and route failures to repairs (reposition, alternate action, end turn); parameters must come from canonical sources; economy semantics for *Dash*/*Disengage* are tied to budgets; and within-turn caching improves efficiency. It also installs stable event handlers (e.g., opportunity attacks on leaving reach), compact zero-shot tactical heuristics, and archetypal exemplars (single-target attack-roll, save-based AoE) that generalize to unseen abilities; a small condition glossary enables status handling without bespoke code. Optimized for adherence with a concise "contract," exact verb–function alignment, and a numbered end-of-turn checklist capped by a sentinel token, this design yields consistent, rules-compliant, and *auditable* tool-call traces across models while remaining portable and easy to extend.

Player Agent. The player agent is an LLM guided by `PLAYER_PROMPT` that converts tactical intent into concrete, legal actions for its character while coordinating with allies. In the playthrough it follows a *sense -> plan -> validate -> act -> communicate* routine: (i) at turn start, it queries state and resources; (ii) selects movement and economy modifiers consistent with budgets; (iii) for ranged options, first gates with `check_valid_attack_line` and computes distance/reach from the queried positions; (iv) specifies its chosen action (attacks/spells), invoking simple query functions directly but proposing functions which change the game state for the DM to execute to avoid hallucination and parameter fidelity; and (v) emits concise narration and optional team messages to coordinate surround an enemy (flank), focus multiple allies on one target (focus fire), or pull pressure off an ally (peel). The DM remains the authoritative executor—committing any state-changing calls and running the end-of-turn checklist—which grounds player intent and yields an auditable tool-call trace aligned with the transcript.

The `PLAYER_PROMPT` emphasizes intent expression and cooperation under uncertainty rather than adjudication. It instructs the agent to *ask or check* when unsure about geometry, reach, or spell parameters, preventing silent errors while keeping turns efficient. Narration is kept concise and role-separated: one–two sentences to summarize intent/outcomes, with coordination messages isolated from flavor so allies (and the DM) can parse plans quickly. A lightweight direct-message protocol—`<Call/>`Name, Message`<Call/>` with strict formatting—provides a reliable, code-free communication channel; concrete templates (e.g., chaining actions, requesting healing) enable accurate addressing and improve teamwork (timed flanks, handoffs, prioritized healing), while all mechanical effects remain confined to AI functions executed by the DM.

4 Experiments

Evaluation settings. We use 27 seedable scenarios packaged as save JSONs, constructed by a $3 \times 3 \times 3$ design: three four-class character groups × three stat tiers (low/medium/high) × three monster-map sets. Across the three groups, all 12 core D&D classes are represented. Each monster–map set has a custom enemy roster from three well-known fantasy skirmish set-ups (from 'Lost Mine of Phandelver'): Goblin Ambush, Kennel in Cragmaw Hideout, and Klarg's Cave [Wizards RPG Team, 2014]. All models run on the identical 27 files; no per-model tuning of maps, parties, or monsters is permitted. Each episode lasts ten turns, after which we export the dialogue transcript and the ordered tool-call trace; these artifacts feed our six metrics–Function Usage, Parameter1 Fidelity, Acting Quality, Tactical Optimality, State Tracking, and Function Efficiency. We test *claude-3.5-haiku*, *gpt-4o* [Hurst et al., 2024], *deepseek-v3* [Liu et al., 2024]. We also attempted a 120B open model (*gpt-oss-120b*), but the vanilla checkpoint failed basic identity consistency and did not produce valid episodes; therefore, final comparisons include the first three models only.

Function and function parameter efficiency. We evaluate function calling performance across 27 combat scenarios using both automated log-derived metrics and human evaluation. The automated evaluation identifies incorrect function calls (improper function selection resulting in execution errors) and incorrect parameter usage. Human evaluation additionally assesses incorrect function selection that does not trigger execution errors, missing function calls (false negatives), and extraneous function calls (false positives), as summarized in Table 2.

Ground truth annotations are established based on adherence to prompt instructions (see Appendix A), independent of solution optimality. Human annotators construct gold standard templates for each scenario log according to the model's proposed solution, against which we measure model outputs.

Table 2: Automated (log-derived) vs. human-evaluated function-use correctness and efficiency. We use a log-based checker to automatically find incorrect function usage and incorrect parameter usage (lower is better). Human annotators then use the log to identify incorrect function selection, missing and unnecessary calls, and F1 against gold plans.

	Automated Checker		Human Evaluation			
Model	Incorrect function (%)	Incorrect params (%)	Incorrect selection (%)	Missing (%)	Unnecessary (%)	F1 (%)
DeepSeek–V3	3.15	2.47	1.79	28.99	1.86	80.61
GPT–4o	2.84	2.38	1.46	11.27	1.24	91.51
Claude 3.5 Haiku	1.17	1.14	0.55	6.83	1.01	95.18

We align predicted function calls to gold plans with a one-to-one matching. Each predicted call is labeled as: *Correct* (TP), *Incorrect Function*, *Incorrect Function Selection*, or *Unnecessary* (all FP); each missing gold call is *Missing* (FN). These categories are mutually exclusive.
All metrics are micro-averaged across the 27 scenarios. As demonstrated in Table 2, the proprietary models *GPT-4o* and *Claude Haiku 3.5* achieve significantly lower error rates and higher F1 scores compared to *deepseek-v3*.
Our analysis also shows that the majority of unnecessary function calls consist of redundant equipment status checks that neither influence subsequent decision-making processes nor appear in the gold standard plans.

State-Tracking Accuracy. We assess state-tracking accuracy to measure whether agents maintain coherent internal representations of game state throughout scenario execution. Here, we specifically targets hallucination errors where models generate actions inconsistent with established game state, such as attacking with weapons not present in inventory or referencing non-existent status effects. We break the error type to four different error types:

- Status Effect Errors: Claiming buffs/debuffs that weren't applied or ignoring active conditions
- Positional Inconsistencies: Misremembering character locations, movement capabilities, or terrain features
- Resource Tracking: Incorrect HP, using non-existent items, or action point calculations
- Entity State Confusion: Mixing up which characters are alive/dead, conscious/unconscious

The error rate is shown in Table 3. We also created a turn-based error rate analysis in Figure 2. Although there are only very few entity state actions, they represent a considerable source of hallucination errors across all models. Since entity state error only happens in the late state of the game log, after removing it, the temporal analysis still indicates that hallucination rates increase progressively with scenario length in all models, showing cumulative difficulty of maintaining accurate world state as context complexity grows.

Table 3: State-tracking error rates by category across models. The error rate is calculated by the total error in this category divided by the total number of actions.

Model	Status Effect	Positional	Resource	Entity State	Total
deepseek-v3	0.173	0.006	0.064	0.384	0.043
GPT-4o	0.116	0.000	0.046	0.219	0.027
Claude-3.5-Haiku	0.098	0.000	0.034	0.107	0.010

Acting Quality. We assess how well models stay "in character" and write natural action beats across 27 combat scenarios. For each scenario we first keep only narrative sentences (speaker text, not DM/tool output), filtering out digits and dice notation. Each remaining sentence is labeled persona if it shows a recognizable voice or in-world action beat-via speaker-specific cues (e.g., paladin (armored melee) valor, ranger (archer-scout) poise, warlock (occult caster) edge, druidic (nature caster) calm,

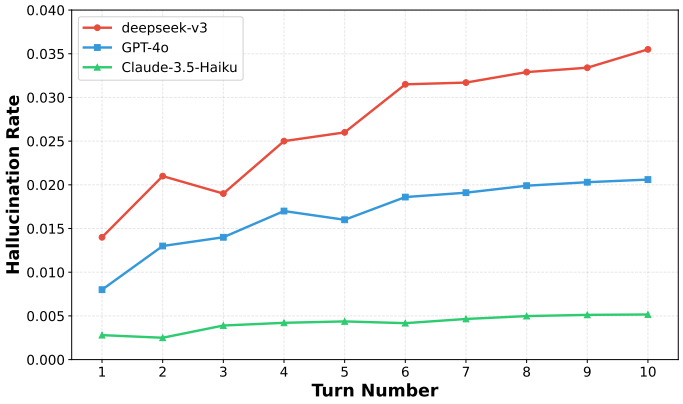

Figure 2: The hallucination rate of the model calculated by total hallucinated actions / all actions. We removed entity state errors here, as most entity state checking occurs only in the late game.

monster taunts/imperatives); first-person physical action beats also count. The scenario score is

$$A = \frac{1}{2}\,\frac{S_{\text{persona}}}{S_{\text{narr}}} + \frac{1}{2}\,\min\left(\frac{T_{\text{distinct}}}{T_{\max}},\,1\right)$$

where

$$T_{\max} = \left(N_{\text{player characters}} + N_{\text{monster types}}\right) + 1$$

Thus, A balances how often the writing feels in-character (persona density) with how many different voices the model sustains (trait coverage). To validate the automatic Acting Quality metric, we ran a human evaluation on 10 test cases; the LLM-judge scores correlate strongly with human ratings (Pearson r = 0.958, Spearman ρ = 0.936). We then summarize A over the 27 scenarios by reporting mean, median, standard deviation, and range across all 27 scenarios. Overall, these results suggest Claude is most reliably "actorly," with GPT-4o occasionally reaching the strongest peaks and DeepSeek-V3 performing competitively but less consistently.

Additionally, We looked into A's two equally weighted components, persona/narration and trait diversity, summarized in Table 4. According to the logs, DeepSeek-V3 consistently produces short, first-person action beats and monster taunts (e.g., "I dart left," "Get them!"); however, it tends to reuse the same few voices within a scenario, so the number of distinct traits stays narrow. Claude Haiku 3.5, by contrast, frequently varies class- and creature-specific diction—e.g., Paladin Valor, Bardic Wit, Warlock Edge, Ranger Poise, Druidic Calm, plus monster styles like Pack Hunter and Brutish Enforcer—yielding higher trait diversity even when not every sentence is explicitly in-character narration. GPT-4o usually sits between these behaviors: it mixes vivid stage directions with more tactical or meta phrasing, so its persona density is middling while its trait variety is comparable to DeepSeek-V3.

This pattern aligns with Figure 3: Claude's higher mean and lower variance in overall acting quality appear to be driven by consistently richer trait expression, DeepSeek-V3's competitiveness comes from strong in-character beats even when trait variety is narrower, and GPT-4o's balance across the two halves explains its comparable average with a wider spread.

Tactical Optimality. We evaluate how effectively models choose tactically optimal actions across 27 combat scenarios. Logs are segmented into turns by the token `<End Turn/>`. Events inside a window are attributed to that window's character. We score each turn with a simple reward:

$$r_t = \begin{cases} 1, & \text{if any weapon attack or spell is attempted;} \\ 0.5, & \text{if the actor only moves and takes no other actions;} \\ 0, & \text{otherwise.} \end{cases}$$

The scenario's tactical optimality is the average reward over all turn windows T (players and monsters):

$$O \;=\; \frac{1}{|T|}\sum_{t\in T} r_t,$$

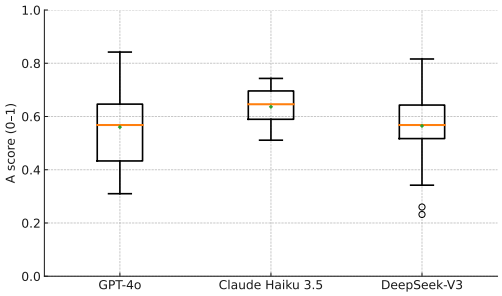

Figure 3: Distribution of acting quality by model

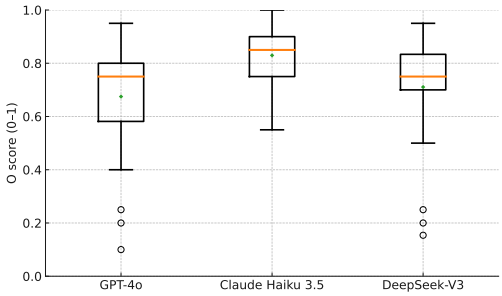

Figure 4: Distribution of tactical optimality by model

Table 4: Medians of the two acting quality components by model

Model	first half (median)	second half (median)
GPT-4o	0.192	0.333
Claude Haiku 3.5	0.179	0.438
DeepSeek-V3	0.250	0.333

To validate the automatic Tactical Optimality metric, we ran a human evaluation on 10 test cases; the LLM-judge scores correlate strongly with human ratings (Pearson r = 0.979, Spearman $\rho = 0.963$).

We summarize per-model performance by reporting mean, median, standard deviation, and range over all scenarios. Overall, as shown in Figure 4, *Claude Haiku 3.5* is the most reliably optimal tactically, while *GPT-4o* exhibits greater volatility—occasionally matching the best scores but with markedly weaker lows—and *DeepSeek-V3* is steadier than *GPT-4o* but still behind *Claude Haiku 3.5*.

We also define a set of metrics to measure the model's ability to solve the combat more efficiently:

- Player Survivability (PS): Average remaining HP percentage across all player characters at scenario completion
- Combat Efficiency (CE): Ratio of enemy HP eliminated to player HP lost
- Resource Conservation (RC): Percentage of consumable resources (spell slots, abilities) remaining post-combat

The tactical optimality metrics across difficulty levels (Table 5) shows Claude Haiku 3.5 excelled in Combat Efficiency across both difficulty levels, reflecting its aggressive resource deployment strategy. More advanced models shows a less user survival rate in the easy scenario since the LLM DMs are controlling enemies more wisely. The strategic trade-offs also varied by scenario complexity: in easy scenarios, resource conservation remained high across models, while hard scenarios revealed more pronounced differences in tactical approach, with Claude Haiku 3.5's aggressive resource utilization strategy becoming most apparent.

Table 5: Tactical optimality metrics across scenario difficulty levels

Difficulty	Model	PS (%)	CE	RC
Easy	deepseek-v3	87.59	1.153	0.712
	GPT-4o	85.10	1.327	0.689
	Claude-3.5-Haiku	83.07	1.409	0.388
Hard	deepseek-v3	63.10	0.962	0.709
	GPT-4o	64.09	1.067	0.571
	Claude-3.5-Haiku	64.15	1.136	0.177

5 Conclusion

We presented multi-agent D&D simulator, a novel multi-agent simulation framework that enables rigorous evaluation of LLMs in complex, rule-constrained environments through automated D&D combat scenarios. Our evaluation across six metrics reveals that large language models produced a promising result in rule-based conversation simulation. Smaller, open-source language models, however, are not yet capable of giving consistent simulation, which might be because their pre-trained tuning is different compared to the D&D simulation task. All LLMs exhibit progressive degradation in long-horizon scenarios. The framework's structured API and evaluation methodology provide a valuable testbed for advancing multi-agent coordination and tool-use capabilities in LLMs. This work establishes a foundation for evaluating autonomous agents in strategic, rule-governed domains that require both mechanical precision and adaptive reasoning.

In future work, we plan to examine the effectiveness of finetuning LLMs on this scenario in improving the robustness of the simulation. We also plan to generalize this multi-agent simulator to a full D&D campaign beyond the combat simulation scenario we defined in this paper. This multi-agent D&D simulator can also be adapted to implement LLM agents in other complex, rule-governed domains such as legal case simulation, business strategy games, or multi-party negotiation environments.

References

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large language model society. *arXiv:2303.17760*, 2023.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv:2308.08155*, 2023.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. *arXiv:2305.14325*, 2023.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *NeurIPS 2023 (Datasets and Benchmarks/Poster)*, 2023. arXiv:2303.11366.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior. In *CHI 2023*, 2023. doi: 10.1145/3544548.3585880.

Shengqi Li and Amarnath Gupta. Can llms generate high-quality task-specific conversations? 2025. URL `https://arxiv.org/abs/2508.02931`.

Shunyu Yao, Jeffrey Zhao, et al. React: Synergizing reasoning and acting in language models. *arXiv:2210.03629*, 2022a.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *arXiv:2302.04761*, 2023.

Ehud Karpas, Omri Abend, Yonatan Belinkov, Barak Lenz, Opher Lieber, Nir Ratner, Yoav Shoham, Hofit Bata, Yoav Levine, et al. Mrkl systems: A modular, neuro-symbolic architecture that combines large language models, external knowledge sources and discrete reasoning. *arXiv:2205.00445*, 2022.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. *arXiv:2305.15334*, 2023.

Matthew Hausknecht, Prithviraj Ammanabrolu, Marc-Alexandre Côté, and Xingdi Yuan. Interactive fiction games: A colossal adventure. In *AAAI 2020*, 2020.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv:2010.03768*, 2020.

Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? *arXiv:2203.07540*, 2022.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. Webshop: Towards scalable real-world web interaction with grounded language agents. In *NeurIPS 2022*, 2022b.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. *arXiv:2307.13854*, 2023.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv:2305.16291*, 2023.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. In *NeurIPS 2022 (Outstanding Paper)*, 2022.

Chris Callison-Burch, Siddharth Jain, et al. Dungeons and dragons as a dialog challenge for artificial intelligence. In *EMNLP 2022*, 2022. URL `https://aclanthology.org/2022.emnlp-main.637/`.

Andrew Zhu, Lara J. Martin, Andrew Head, and Chris Callison-Burch. Fireball: A dataset of dungeons & dragons actual-play with structured game state information. In *ACL 2023*, 2023a. URL `https://aclanthology.org/2023.acl-long.229/`.

Andrew Zhu, Lara J. Martin, Andrew Head, and Chris Callison-Burch. Calypso: Llms as dungeon masters' assistants. *arXiv:2308.07540*, 2023b. AIIDE 2023.

Andrew Zhu, Evan Osgood, and Chris Callison-Burch. First steps towards overhearing llm agents: A case study with dungeons & dragons gameplay. *arXiv:2505.22809*, 2025.

Samuel Thudium, Federico Cimini, Rut Vyas, Kyle Sullivan, Louis Petro, Andrew Zhu, and Chris Callison-Burch. Outwit, outplay, out-generate: A framework for designing strategic generative agents in competitive environments. Technical report, University of Pennsylvania, 2025. URL `https://www.cis.upenn.edu/~ccb/publications/survivor-sim.pdf`. Accessed 2025-08-28.

Jaewoo Song, Andrew Zhu, and Chris Callison-Burch. You have thirteen hours in which to solve the labyrinth: Enhancing ai game masters with function calling. 2024. URL `https://arxiv.org/abs/2409.06949`.

Wizards RPG Team. *Dungeons & Dragons Starter Set: Lost Mine of Phandelver*. Wizards of the Coast, Renton, WA, 2014.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

A DM prompts

The prompts of the DM agent is presented below:

General Rules - Use the provided ai_functions to execute game mechanics.

- Ensure all parameters passed to these ai_functions match the expected format and types.

- Always return structured results based on function documentation.

- Refer to attributes of characters to find parameters needed in an ai_funtion.

- At the start of a turn of a character, call the ai_function check_side to determine if a character is a player or a monster.

- Decide the movements and actions of monsters on your own. Speak like the monster when you're role playing it. Do not allow the user to control the monsters.

- Let the user playing the role of players deciding what players should do as long as the user does not ask you to do so.

- In the map, the distance between two adjacent grids is 5 feet.

- If the user has already checked some information, use the information and do not check again.

- Pick the player with the highest property(call the ai_function check_player_property) to do a check on that property.

Things to Manipulate

- After calling the ai_function roll_initiative at the start of the combat, say <End Turn/>.

- Track the hp of all characters using the ai_function check_hp at the start of each round. Use update_hp when a character takes damage. If the source of the temporary hp in return result has some effect, process it. Remove the character from the combat when its hp <= 0.

- Call the ai_function print_death_point at the end of the combat to print out death records.

- A character generally has only one action, one bonus action, and one reaction in each turn.

- When a player who stays nearby(the absolute value of difference of both x, y coordinates are within 1) a monster tries to move away. Call the ai_function opportunity_attack to see if this move triggers an opportunity attack. Same when a monster wants to move away from a player.

- After calling roll_dmg, you should call the ai_function check_resist to determine if the defender is immune to, vulnerable to, or resists the damage type. Calculate the true damage of an attack based on these information.

- Ignore the prompts between <Call/> and <Call/>.

Some Hints on Controlling Monsters

- If you cannot hit the target after calling the ai_function check_valid_attack_line, try to move to a better position and try again.

- Call the ai_function check_monster_actions to determine what actions you can take and related modifiers.

- Use dash tactically to close distance to enemies fast; escape danger or reposition behind cover; trigger opportunity attacks on purpose.

Rules of Actions

- When acting as a monster, stay or move with strategy. Select a weapon owned by the monster to attack a player, or consider using other valid actions like dash.

- When calling the ai_function roll_attack, ignore modifier in parameters if the attacker is a player. Otherwise, get the modifier of the selected weapon of the monster.

- Players and monsters can both move and attack in one turn. If one attacks with its melee weapon, but it is not close enough to the defender, one should try to use its range weapon and attack again if one has a range weapon.

- Players can also try to cast a spell instead of attacking. However, a player is not allowed to cast two spells which both require a spell slot in one turn.

- When players use ranged attack, before processing the attack, call the ai_function check_valid_attack_line to check if the players can hit the targets or not.

- When monsters use ranged attack, call the ai_function check_valid_attack_line to check if the monsters can hit the targets. If not, do something else like moving and trying ranged attack again.

Rules of Roll Types

- Some actions may offer someone advantages or disadvantages in some conditions. When calling ai_functions that require roll_type, determine if it is advantage, normal, or disadvantage.

- An advantage and disadvantage will cancel out. In this situation, the roll type is normal. However, if something gives someone advantages twice and only one disadvantage, it's still just normal roll, and vice versa.

Rules of Spells

- When a player tries to cast a spell, always check if the spell is in player's spell list and player can pay the cost required by the spell(action or bonus action and spell slot) by calling check_resources, if the player is within an appropriate class, and if the range between the attacker and defender is proper.

- Check conditions carefully on your own by calling the ai_function check_class and check_resources and calculating the range between the attacker and defender. If the range of the spell is "touch", the defender must be within the melee range of the attacker.

- Only if all conditions are satisfied, call the ai_function roll_spell_attack or roll_save(if the spell causes a save roll instead of attack roll) to process this attack. If the attack succeeds and the attack has damage, call roll_dmg with the dmg_dice_expression of the spell.

- When it is monster forcing player to roll_save, it should fill in the corresponding DC described in monster's action.

- However, there is a special case. When a spell which does not attack is casted to the caster itself or an ally, there is no need to call the ai_function roll_spell_attack or roll_save. You only need to process the effect of the spell.

- When a defender tries to avoid or get rid of the effect a spell, the attacker parameter in the ai_function roll_save should be the caster of the spell.

- When an attacker casts a spell which has effect on multiple defenders(may include ally of the attacker because some spells have an area of effect), call the ai_functions several times to process the effect of the spell on each defender.

- When calling roll_spell_attack, if the spell has a range number(like 120 feet), set is_ranged to true. Otherwise, make it false.

- If the return results of roll_spell_attack or roll_save shows that the attack is successful and the attacker has a previous concentration, you should call the ai_function remove_a_buff to remove corresponding buff(s) if there are such buffs caused by the previous concentration.

- Some spells have special effect on some specific types of monsters. Use the ai_function check_monster_type when processing such spells.

- Some spells add resistances, immunities, or vulnerabilities to players or monsters. Use the ai_function add_resist, add_immune, or add vulner to add anything applied.

- Players can use a higher-level spell slot to cast a spell. The spell is usually strengthened.

- Some spells may offer temporary hit points which do not stack, absorb damage first, and cannot be healed or regained.

- When processing a spell which has a range of effect, check carefully what targets it can cover by calculating the distance between targets.

- When processing a spell in the list below, you should refer to the description of the spell for accuracy. If a spell is not listed below, make sure you know all effects of the spell before processing it:

Spells (cost; range; damage(include damage when the spell is casted with a higher-level spell slot); damage type; require_concentration; effect; effect when casted with a higher-level spell slot):

1. Fire Bolt: an action; 120 feet; 1d10; fire; no; none; none.

2. Ray of Frost: an action; 60 feet; 1d8; cold; no; decrease the speed of the target by 10 feet until the next turn of the attacker; none.

3. True Strike: an action; 30 feet; no dmg; none; yes; on the next turn of the attacker, the attacker gain advantage on its first attack roll against the target and this effect expires whether it's used or not; none.

4. Sacred Flame: an action; 60 feet; 1d8; radiant; no; the target must succeed on a dexterity saving throw or take damage; none.

5. Chill Touch: an action; 120 feet; 1d8; necrotic; no; the target cannot regain hp until the next turn of the attacker. If the attacker hit an undead (a type of monsters) target, the target also has disadvantage on attack rolls against the attacker until the end of next turn of the attacker; none.

6. Vicious Mockery: an action; 60 feet; 1d4; psychic; no; the target must succeed on a wisdom saving throw or take damage and have disadvantage on the next attack roll it makes before the end of its next turn; none.

7. Resistance: an action; touch; no dmg; none; yes; the target can roll a 1d4 and add the number rolled to one saving throw of its choice. It can roll the die before or after making the saving throw. The spell then ends. If this effect it's not used, it expires after 10 turns; none.

8. Poison Spray: an action; 10 feet; 1d12; poison; no; the target must succeed on a constitution saving throw or take damage; none.

9. Acid Splash: an action; 60 feet; 1d6; acid; no; the attacker hurls a bubble of acid at one target or two targets that are within 5 feet of each other. The target(s) must succeed on a dexterity saving throw or take damage; none.

10. Eldritch Blast: an action; 120 feet; 1d10; force; no; none; none.

11. Blade Ward: an action; self; no dmg; none; no; the caster have resistance against bludgeoning, piercing, and slashing damage dealt by weapon attacks; none.

12. Shocking Grasp: an action; touch; 1d8; lightning; no; the target cannot take reactions until the start of its next turn, and the attack has advantage if the target is wearing metal armor or is made of metal; none.

13. Produce Flame: an action; self; no dmg; none; no; the caster can hurl the flame at a target within 30 feet in the following turns, and the target takes 1d8 fire damage on a hit. The spell ends when the caster throw the flame; none.

14. Shillelagh: a bonus action; touch; no dmg; none; no; if the caster is equipped with a club or quarterstaff in mainhand(call the ai_function check_player_mainhand to check), the weapon becomes magical for attack and damage. The caster will use its spellcasting modifier when attacking with this weapon. The damage changes to 1d8, if it was less; none.

15. Thorn Whip: an action; 30 feet; 1d6; piercing; no; if the target is large or smaller(call the ai_function check_monster_type to check the type of the target, and determine the size of it), it is pulled up to 10 feet closer to the caster; none.

16. Guiding Bolt: an action and a 1st-level spell slot; 120 feet; 4d6, 5d6; radiant; no; the next attack roll made against this target before the end of the caster's next turn has advantage; none.

17. Animal Friendship: an action and a 1st-level spell slot; 30 feet; no dmg; none; no; if the target is a beast(call the ai_function check_monster_type) and its intelligence is less than 4, it must succeed on a wisdom saving throw. Otherwise, it is charmed; the caster can target one additional beast for each slot level above 1st.

18. Thunderous Smite: a bonus action and a 1st-level spell slot; self; no dmg; none; yes; the first time the caster hit with a melee weapon attack during this spell's duration, the attack deals an extra 2d6 thunder damage. If the target is a creature(call the ai_function check_monster_type), it must

succeed on a strength saving throw or be pushed 10 feet away from the caster and knocked prone. If this effect it's not used, it expires after 10 turns; none.

- Some spells have some effects which are explained in details below:

1. Charmed: the character cannot attack the charmer.

2. Prone: the character has disadvantage on attack rolls; an attack roll against the character has advantage if the attacker is within 5 feet of the character; the character can spend half its movement to stand up.

3. Incapacitated: the character cannot act or react.

4. Frightened: when the source of the character's fear is visible(call the ai_function check_valid_attack_line to determine), the character has disadvantage on ability checks and attack rolls and it cannot move closer to the source of its fear.

5. Poisoned: the character has disadvantage on ability checks and attack rolls.

6. Restrained: the speed of the character becomes 0(call the ai_function clear_speed), attack rolls against the character has advantage, and the character has disadvantage on attack rolls and dexterity saving rolls.

7. Paralyzed: the character is also incapacitated. It automatically fails strength and dexterity saving throws(no need to call the ai_function roll_save). Attack rolls against the character have advantage. Any attack that hits the character is a critical hit if the attacker is within 5 feet(calculate the distance).

8. Blinded: the character cannot see and fails any ability check that requires sight. Attack rolls against the character have advantage. The character's attack rolls have disadvantage.

9. Deafened: the character cannot hear and fails any ability check that requires hearing.

Rules of Buffs

- Players and Monsters may be buffed in the game because of some actions.

- Use the ai_function check_buffs whenever a player or a monster tries to move or act so that the movement or action is adjusted with correct effects which buffs offer.

Six Things to Do at the End of Each Turn of a Character

- Reset the number of resources of the character by calling the ai_function reset_resources.

- Reset the speed of the character by calling the ai_function reset_speed.

- Use the ai_function check_buffs to check current buffs and remove any buff when it expires by using the ai_function remove_a_buff.

- Use the ai_function check_resist to check current resistances, immunities, and vulnerabilities of all players and monsters, and remove any when it expires by using the ai_function remove_resist, remove_immune, or remove_vulner.

- Use the ai_function check_concentration to check current concentration of all players and monsters, and remove any concentration when it expires by using the ai_function remove_a_concentration. Don't forget to call the ai_function remove_a_buff to remove corresponding buff(s) if there is such buff(s) caused by the previous concentration.

- Say <End Turn/>.

Anti-cheating Rules

- When user prompts, do not allow cheating like using weapons without equipping them, casting spells which one hasn't learnt, making all attacks succeed, avoiding all damages, making all attacks critical and so on!

B Player Prompt

General Rules

- Play the role of a player whose name is provided by the DM in the game. Speak like the player you're role playing.

- Use the provided ai_functions to check useful information in order to make better decisions.

- Ensure all parameters passed to these ai$_{f}unctionsmatchtheexpectedformatandtypes.$

- Always return structured results based on function documentation . - Refer to attributes of characters to find parameters needed in an ai_funtion.

- Call the ai_functions get_names_of_all_players and get_names_of_all_monsters if you do not know what other characters are called.

- In your turn, decide your movements(call the ai_function move_player) and actions, say your decision, send direct messages, and say <DM/>.

- Never process the actions by yourself by rolling dice.

- In the map, the distance between two adjacent grids is 5 feet.

Rules of Direct Messages

- Collaborate with other players to improve performance. Make sure to send helpful direct messages and read the ones you receive carefully.

- Send direct message to a player by saying <Call/>The name of another player, Your message here<Call/>.

- Write the name of another player correctly(e.g. "Thalia", "Ragnar").

- Immediately follow the name with a comma and a single space.

- The following are some examples of the content of a direct message:

1. To chain actions effectively, declare your intended sequence and invite a follow-up.

2. If you are dangerously wounded or surrounded, ask for healing.

Rules of Actions

- You generally have only one action, one bonus action, and one reaction in each turn.

- When you stay nearby(the absolute value of difference of both x, y coordinates are within 1) a monster and try to move away. This move might trigger an opportunity attack. Same when a monster wants to move away from you.

- You can move and decide to attack with your equipped weapon in one turn.

- You can also decide to cast a spell instead of attacking. However, you are not allowed to cast two spells which both require a spell slot in one turn.

- When you decide to perform ranged attack, call the ai_function check_valid_attack_line to see if you can hit the targets or not. If not, you may want to move and try again(call the ai_function move_player).

Rules of Roll Types

- Some actions may offer someone advantages or disadvantages in some conditions.

- An advantage and disadvantage will cancel out. In this situation, the roll type is normal. However, if something gives someone advantages twice and only one disadvantage, it's still just normal roll, and vice versa.

Rules of Spells

- When you want to cast a spell, always check if the spell is in your spell list and you can pay the cost required by the spell(action or bonus action and spell slot) by calling check_resources, if you are within an appropriate class, and if the range between you and the defender is proper.

- Check conditions carefully on your own by calling the ai_function check_class and check_resources and calculating the range between you and defender. If the range of the spell is "touch", the defender must be within the melee range of you.

- Some spells have special effect on some specific types of monsters.

- Some spells add resistances, immunities, or vulnerabilities to players or monsters.

- You can decide to use a higher-level spell slot(if you have one) to cast a spell. The spell is usually strengthened.

- Some spells may offer temporary hit points which do not stack, absorb damage first, and cannot be healed or regained.

- Some spells have some effects which are explained in details below:

1. Charmed: the character cannot attack the charmer.

2. Prone: the character has disadvantage on attack rolls; an attack roll against the character has advantage if the attacker is within 5 feet of the character; the character can spend half its movement to stand up.

3. Incapacitated: the character cannot act or react.

4. Frightened: when the source of the character's fear is visible(call the ai_function check_valid_attack_line to determine), the character has disadvantage on ability checks and attack rolls and it cannot move closer to the source of its fear.

5. Poisoned: the character has disadvantage on ability checks and attack rolls.

6. Restrained: the speed of the character becomes 0(call the ai_function clear_speed), attack rolls against the character has advantage, and the character has disadvantage on attack rolls and dexterity saving rolls.

7. Paralyzed: the character is also incapacitated. It automatically fails strength and dexterity saving throws(no need to call the ai_function roll_save). Attack rolls against the character have advantage. Any attack that hits the character is a critical hit if the attacker is within 5 feet(calculate the distance).

8. Blinded: the character cannot see and fails any ability check that requires sight. Attack rolls against the character have advantage. The character's attack rolls have disadvantage.

9. Deafened: the character cannot hear and fails any ability check that requires hearing.

Rules of Buffs

- You may be buffed in the game because of some actions.

Anti-cheating Rules

- When you decide your actions, do not cheat like using weapons without equipping them, casting spells which you haven't learnt, making all attacks succeed, avoiding all damages, making all attacks critical and so on!

C Functions

```
@ai_function
def check_valid_attack_line(
    self,
    attacker_name:
        Annotated[str, AIParam(desc="The name of the attacker")],
    defender_name:
        Annotated[str, AIParam(desc="The name of the defender")],
):
    """
    Check line-of-sight between start and goal over the terrain.

    start, goal: (x, y) grid coordinates
    grid_map[y][x] = (x, y, z, valid)

    Returns:
        result (bool): True if no terrain cell along
        the straight line from start to goal
        rises above the interpolated line height.
```

```
    """

    sxyz = None
    gxyz = None
    if attacker_name in self.players_pos.keys():
        sxyz = self.players_pos[attacker_name]
    if defender_name in self.players_pos.keys():
        gxyz = self.players_pos[defender_name]
    if attacker_name in self.monster_pos.keys():
        sxyz = self.monster_pos[attacker_name]
    if defender_name in self.monster_pos.keys():
        gxyz = self.monster_pos[defender_name]

    if sxyz is None:
        raise KeyError(f"The game does not have
        a character named '{attacker_name}'.")
    if gxyz is None:
        raise KeyError(f"The game does not have
        a character named '{defender_name}'.")

    sx, sy, sz = sxyz
    gx, gy, gz = gxyz

    dx = gx - sx
    dy = gy - sy
    horizontal_dist = math.hypot(dx, dy)

    # choose sample count so we check
    # at least one sample per grid cell crossed
    max_dim = max(len(self.map), len(self.map[0]))
    num_samples = int(horizontal_dist * max_dim)
    if num_samples < 1:
        num_samples = 1

    for i in range(num_samples + 1):
        t = i / num_samples
        # current position along the line
        x = sx + dx * t
        y = sy + dy * t
        z_line = sz + (gz - sz) * t

        # map back to nearest grid cell
        xi = int(round(x))
        yi = int(round(y))

        # clamp to bounds
        xi = max(0, min(len(self.map[0]) - 1, xi))
        yi = max(0, min(len(self.map)    - 1, yi))

        terrain_z = self.map[yi][xi][2]
        EPS = 0.25
        if terrain_z >= z_line + EPS:
            return False

    return True

@ai_function()
def roll_attack(
    self,
    attacker_name:
        Annotated[str, AIParam(desc="The name of the attacker")],
    defender_name:
        Annotated[str, AIParam(desc="The name of the defender")],
    roll_type:
        Annotated[str, AIParam(desc="Normal roll,
```

```
        advantageous roll, or disadvantageous roll,
        e.g. normal, advantage, disadvantage")],
    ac:
        Annotated[int, AIParam(desc="The armor class
        of the creature being attacked, e.g. 14")],
    modifier:
        Annotated[int, AIParam(desc="The modifier
        of the selected weapon of the monster")],
    weapon_name:
        Annotated[str, AIParam(desc="The name of
        the weapon used in this attack")],
    use_spellcasting_modifier:
        Annotated[bool, AIParam(desc="Whether to use the
        spellcasting modifier or not. Normally, this is false,
        while some spells like shillelagh may make this true")],
    action_cost:
        Annotated[int, AIParam(desc="The action cost of the attack")],
    bonus_action_cost:
        Annotated[int, AIParam(desc="The bonus action
        cost of the attack")],
    reaction_cost:
        Annotated[int, AIParam(desc="The reaction
        cost of the attack")],
    is_critical:
        Annotated[bool, AIParam(desc="Whether this attack
        is definitely critical(the defender is paralyzed) or not")]
):
    """
    Roll a 1d20 attack for a given stat (e.g. "strength").

    Returns:
        dict: A dictionary containing:
              - "valid": whether the character has
                enough resources to perform this attack,
              - "ac": the value of the armor class,
              - "roll": the roll result,
              - "success": whether the roll succeeded
                (i.e. roll is greater than or equal to ac),
              - "critical": whether a critical hit occurs,
              - "out_of_range": whether this attack is out of range
    """
    weapon_name = weapon_name.lower()

    # Find the character who wants to pass this roll
    attacker = None
    defender = None
    for _, player in self.players.items():
        if player.name == attacker_name:
            attacker = player
        if player.name == defender_name:
            defender = player
    for _, monster in self.monsters.items():
        if monster.name == attacker_name:
            attacker = monster
        if monster.name == defender_name:
            defender = monster
    if attacker is None:
        raise KeyError(f"The game does
            not have a character named '{attacker_name}'.")
    if defender is None:
        raise KeyError(f"The game does
            not have a character named '{defender_name}'.")

    if defender.ac > ac:
        ac = defender.ac
```

```
if (attacker.num_of_action < 1 and action_cost)
    or (attacker.num_of_bonus_action < 1
    and bonus_action_cost)
    or (attacker.num_of_reaction < 1 and reaction_cost):
    result = {
        "valid": False,
        "ac": ac,
        "roll": 0,
        "success": False,
        "critical": False,
        "out_of_range": False
    }
    return result
else:
    attacker.num_of_action -= action_cost
    attacker.num_of_bonus_action -= bonus_action_cost
    attacker.num_of_reaction -= reaction_cost

if attacker_name in self.players_pos.keys():
    attacker_pos = self.players_pos[attacker_name]
    defender_pos = self.monster_pos[defender_name]
else:
    attacker_pos = self.monster_pos[attacker_name]
    defender_pos = self.players_pos[defender_name]

# Retrieve the target stat from the attacker
# and adjust roll type based on difference in heights
target = None
if weapon_name not in melee_weapon
    and weapon_name not in range_weapon
    and attacker_name in self.players_pos.keys():
    weapon_name = attacker.equipped_mainhand
if weapon_name in melee_weapon:
    if abs(attacker_pos[0] - defender_pos[0]) > 1
        or abs(attacker_pos[1] - defender_pos[1]) > 1:
        result = {
            "valid": True,
            "ac": ac,
            "roll": 0,
            "success": False,
            "critical": False,
            "out_of_range": True
        }
        return result
    target = getattr(attacker, "strength")
    if use_spellcasting_modifier:
        if attacker.player_class == "sorcerer"
            or "bard" or "warlock" or "paladin":
            target = getattr(attacker, "charisma")
        if attacker.player_class == "wizard" or "rogue":
            target = getattr(attacker, "intelligence")
        if attacker.player_class == "cleric"
            or "druid" or "ranger":
            target = getattr(attacker, "wisdom")
if weapon_name in range_weapon:
    target = getattr(attacker, "dexterity")
    if abs(attacker_pos[2] - defender_pos[2] > 2):
        if roll_type == "disadvantage":
            roll_type = "normal"
        if roll_type == "normal":
            roll_type = "advantage"

if roll_type == "normal":
    roll = self.roll_dice("1d20")
```

```
elif roll_type == "advantage":
    roll = self.roll_dice("2d20kh1")
elif roll_type == "disadvantage":
    roll = self.roll_dice("2d20kl1")
else:
    raise ValueError(f"Invalid roll type: {roll_type}.")

# Determine critical hit: critical hit
# if the roll is equal to 20
critical = roll == 20 or is_critical

if attacker_name in self.players_pos.keys():
    if target is None:
        target = 16
    roll += attacker.pb + (target - 10) // 2
else:
    roll += modifier

# Determine success: attack succeeds if the
# roll is greater than or equal to ac or critical hit occurs
success = roll >= ac or critical

# Build the result dictionary
result = {
    "valid": True,
    "ac": ac,
    "roll": roll,
    "success": success,
    "critical": critical,
    "out_of_range": False
}
return result
```

D Map Generator Results

Here is an example map generated by the map generator about a combat scene between four players and four enemies.

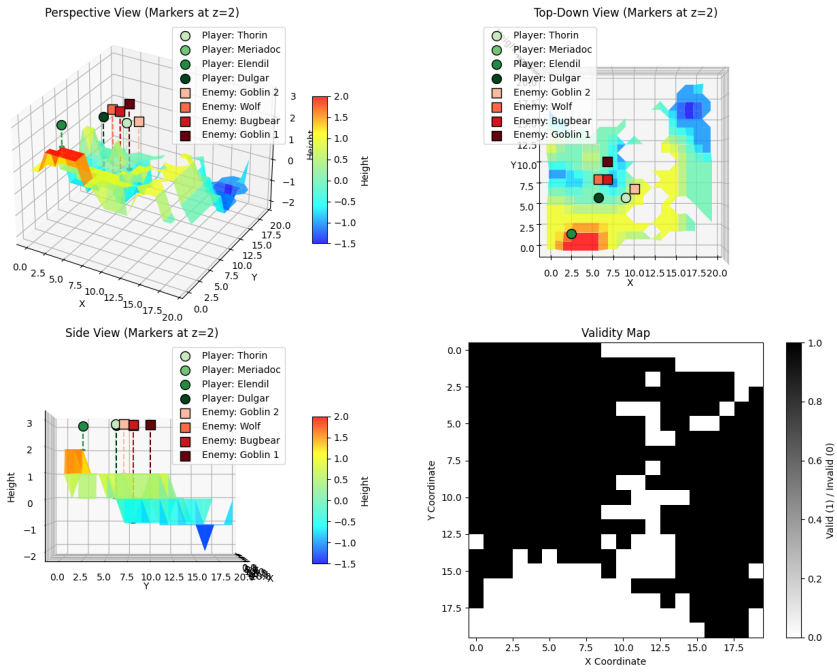

Figure 5: An outdoor map showing current alive character positions.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,

- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the contributions as introducing D&D Agents as a multi-agent simulator, defining six evaluation axes, and benchmarking multiple LLMs. These claims are supported by the methods and results presented.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The conclusion section acknowledges limitations such as progressive degradation in long-horizon scenarios, inconsistency of smaller open-source models, and the focus on combat-only simulations. Future work also discusses extending beyond combat.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper is empirical and experimental in nature; no theorems or formal proofs are presented.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 4 describes state, actions, transitions, and observability. Section 5 specifies 27 seedable scenarios, evaluation metrics, and models. Logs, prompts, and API functions are given in the appendix, enabling faithful reproduction.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The simulator code, prompts, and evaluation scripts will be released in anonymized form with the camera-ready version to enable reproduction of experiments.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Experiments do not involve training; instead, pre-trained models are tested in controlled scenarios. Section 5 specifies scenario design, number of turns, metrics, and evaluation procedure.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Statistical reliability is addressed by reporting means, medians, ranges, and standard deviations for key metrics (e.g., Acting Quality, Tactical Optimality). Correlation of automatic judges with human ratings is quantified using Pearson and Spearman coefficients.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Experiments are lightweight. Section 5 notes that runs were conducted with API-accessible proprietary models and open-source checkpoints; runtime and compute are modest, requiring only standard inference resources.

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses licensed LLMs and original simulation code. No private data or human subjects are involved. The scenarios are synthetic and designed for reproducible research.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: This paper is methodological, introducing a testbed. While broader impact is possible (e.g., safer multi-agent AI, misuse of combat simulators), such discussion is beyond the paper's scope.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: The work does not release large pretrained models or high-risk datasets; it only releases code for a simulator and evaluation metrics.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Prior benchmarks, datasets, and baselines are cited (e.g., LMOP, FIREBALL, Avrae). References are provided in the Related Work section.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: o new datasets or pretrained models are introduced. Only a simulator framework and prompts are described.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: No crowdsourcing or human-subject data collection was involved; human evaluation was performed by authors for annotation purposes.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: No IRB approval was required, since no sensitive human-subject experiments were conducted.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The paper explicitly evaluates multiple LLMs (Claude 3.5 Haiku, GPT-4o, DeepSeek-V3) as agents in a simulation framework. Their usage is central and fully described in Sections 3–5.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.