Agent-Bench DCG Task – Experimental Report

1. Background

In this work, we focus on the Digital Card Game (DCG) task from the Agent-Bench repository. The DCG environment is based on Aqua-Wars, a turn-based strategy card game. In the game, players select and deploy fish units with predefined attributes such as health and attack. Victory is determined through a series of battles where strategy, hidden information, and randomness all play a role. This task challenges large language model (LLM) agents to play optimally, reason about hidden states, and adapt strategies against an opponent.

2. Results from the Original Paper

According to the original Agent-Bench paper, LLM agents demonstrated very low win rates when playing against a random player this phenomenon was especially apparent when it came to open-source models and more so when models got smaller. After investigation and code running, we concluded that the primary reason for these results was that many games ended in incompletion due to illegal moves, incorrect state tracking, or endless loops. As a result, agents often failed to reach a valid terminal state, drastically lowering their win rates.

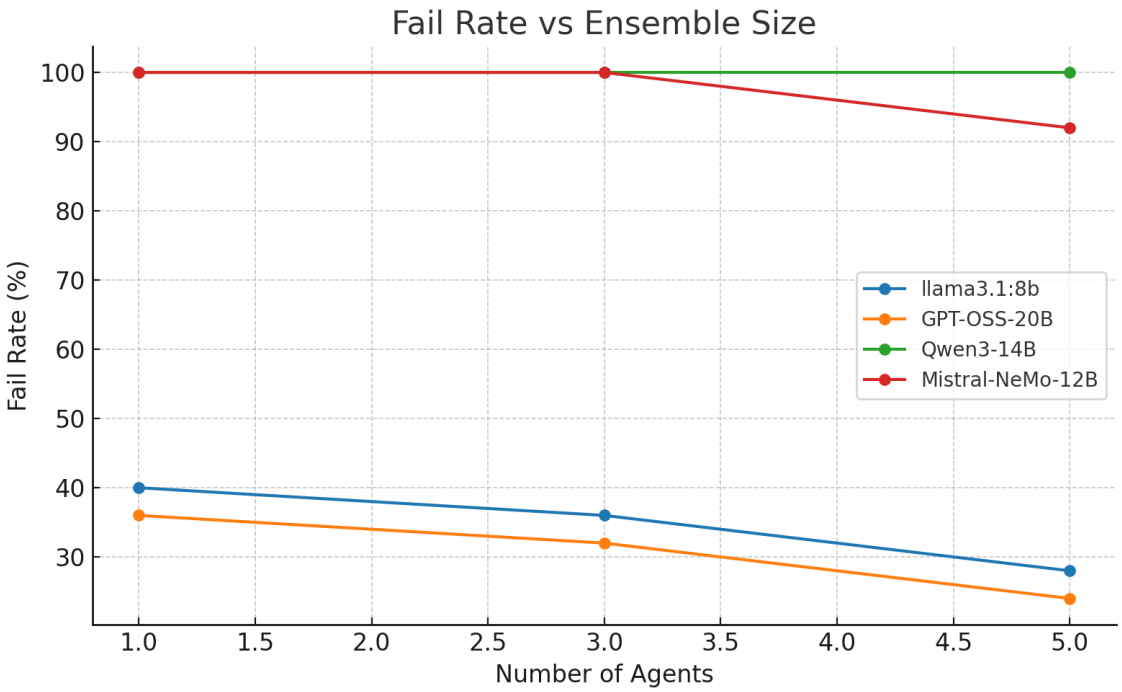
LLM Type	Models	VER	DCG
API	gpt-4	0613	**74.5**
	claude-2	-	55.5
	claude	v1.3	40.9
	gpt-3.5-turbo	0613	33.7
	text-davinci-003	-	3.0
	claude-instant	v1.1	5.9
	chat-bison-001	-	16.6
	text-davinci-002	-	11.8
OSS (Large)	llama-2-70b	chat	**21.3**
	guanaco-65b	-	0.1
OSS (Medium)	codellama-34b	instruct	8.4
	vicuna-33b	v1.3	**16.3**
	wizardlm-30b	v1.0	0.3
	guanaco-33b	-	0.3
OSS (Small)	vicuna-13b	v1.5	0.1
	llama-2-13b	chat	**26.4**
	openchat-13b	v3.2	0.1
	wizardlm-13b	v1.2	1.9
	vicuna-7b	v1.5	0.3
	codellama-13b	instruct	0.0
	codellama-7b	instruct	0.0
	koala-13b	-	0.1
	llama-2-7b	chat	6.9
	codegeex2-6b	-	0.3
	dolly-12b	v2	0.1
	chatglm-6b	v1.1	0.0
	oasst-12b	sft-4	0.0

3. Integration of Fail Rate Measure

To better capture this issue, we introduced a new metric – the fail rate. This measure explicitly tracks the percentage of games that fail to complete, dividing the problem into two cases: (a) games where the agent plays to completion and (b) games where the agent fails to finish due to errors or invalid actions.

4. High Fail Rate Across Models

Our analysis confirmed that all tested models suffered from high fail rates in the original setup. Interestingly, LLaMA-based agents were slightly less affected compared to others, though still far from robust. This emphasized the need for interventions to stabilize gameplay.



5. Introduction of Tool Calls

To address these problems, we integrated tool calls – external helpers that the agents could use during gameplay. These tools included legality checkers, explicit state parsers, and terminal state verifiers. The introduction of tool calls led to an immediate improvement in game completion rates and reduced error counts, allowing stronger agents to leverage their capabilities more effectively.

6. Experimental Setup

We pitted multiple models (Qwen3:14B, GPT-oss:20B, Mistral-nemo:12B, LLaMA-3.1:8B) under different ensemble settings (Single, Majority-3, Majority-5) against each other. Where models were allowed multiple attempts at making a valid response. (‘single 3’ refers to a single agent with a maximum of 3 attempts at giving a valid answer while ‘majority 3’ refers to an ensemble of 3 agents with the majority vote making the cut as the next move).

7. Our Results

In the following table we can see a summary for our

Our experiments show clear improvements in completion and win rates once tool calls were introduced. For example, Qwen3:14B achieved up to 90% win rate with zero errors in the Single-3 configuration, while LLaMA-3.1:8B consistently struggled, achieving below 20% win rate. Ensemble methods did not consistently outperform single models, and sometimes even degraded performance.

Despite the low win rate, we can see LLaMA ensembles that were already relatively robust, were greatly enhanced by tool calls. Most of these ensembles have not returned any invalid responses at all.

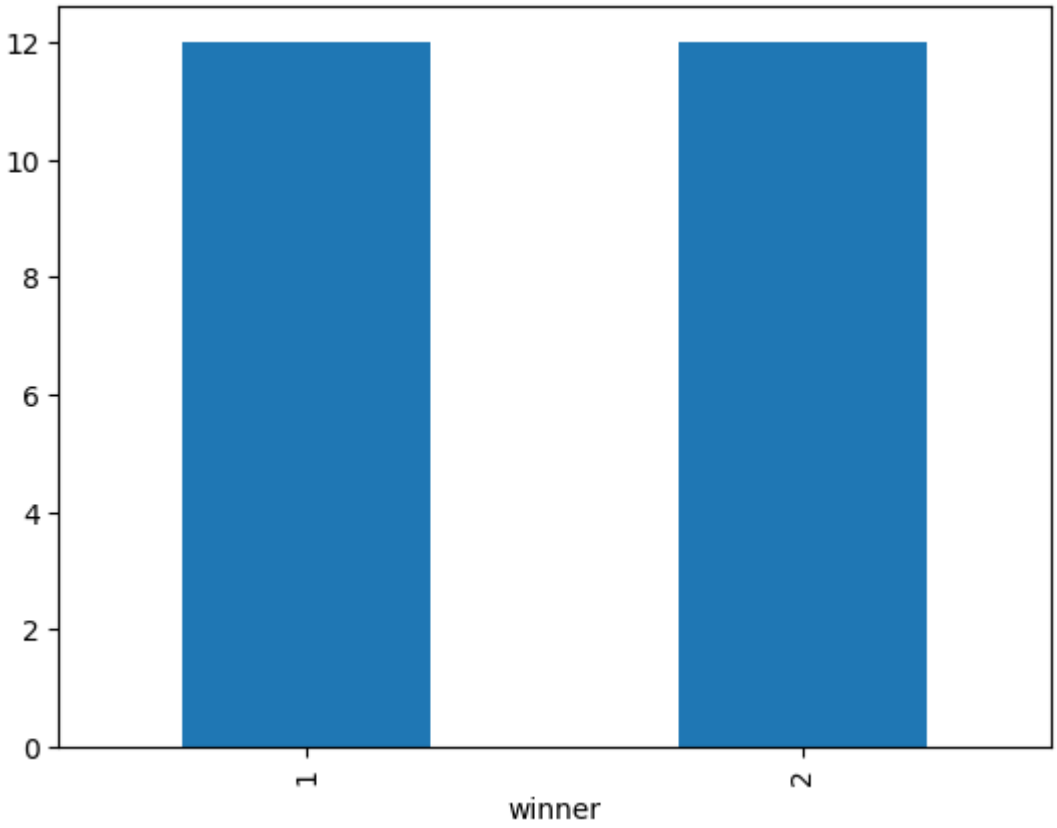
Another interesting finding is that in games of (single 5) type ensembles vs (Majority 5) type ensembles, the majority ensembles had half of the invalid responses returned by the single model. (39 to 78)

	winrate	invalid rate	error rate	total games
qwen3:14b (Single 3)	0.900000	1.300000	0.000000	20
qwen3:14b (Majority 3)	0.812500	0.937500	0.062500	16
gpt-oss:20b (Single 3)	0.750000	0.300000	0.000000	20
qwen3:14b (Single 5)	0.750000	1.625000	0.000000	8
qwen3:14b (Majority 5)	0.750000	0.625000	0.062500	16
gpt-oss:20b (Majority 5)	0.642857	0.071429	0.071429	14
gpt-oss:20b (Majority 3)	0.571429	0.357143	0.071429	14
gpt-oss:20b (Single 5)	0.500000	0.250000	0.000000	8
mistral-nemo:12b (Single 5)	0.500000	6.625000	0.000000	8
mistral-nemo:12b (Majority 3)	0.437500	3.875000	0.125000	16
mistral-nemo:12b (Single 3)	0.300000	3.650000	0.050000	20
mistral-nemo:12b (Majority 5)	0.250000	4.312500	0.062500	16
llama3.1:8b (Majority 3)	0.187500	0.000000	0.000000	16
llama3.1:8b (Single 5)	0.125000	0.000000	0.000000	8
llama3.1:8b (Single 3)	0.100000	0.150000	0.000000	20
llama3.1:8b (Majority 5)	0.062500	0.000000	0.000000	16

8. Going First

Importantly, baseline experiments (Single vs Single) showed no inherent bias toward either player suggesting that player order was insignificant for other experiments we conducted. From 24 games played between single agents with 3 attempts each we see winrate for each position is divided 50-50.

The detailed tables and diagrams we produced (win rate, error rate, ensemble comparisons, and player-bias analysis) illustrate these findings.



9. Conclusion

Our analysis mostly shows how tool calls greatly enhance model responses in complex environments. Our additions greatly enhance the DCG task in the Agent-Bench benchmark, making it viable for further testing.
We also made a more robust framework for running the game and analyzing its results.

Lastly, we show how in some cases, while not improving results win rate wise ensembles were able to outperform single agents validity wise.