Punching above its weight(s)

Benchmarking Large Language Models' reasoning ability using Schnapsen card-playing platform.

Aleksey M. Martynyuk[1,2] and Tommaso Zambelli Franz[1,3]

[1] Vrije Universiteit Amsterdam, De Boelelaan 1105, 1081 HV, Amsterdam, The Netherlands
[2] 2817134
[3] 2825713

Abstract. We are proposing an approach for testing and benchmarking reasoning abilities of Large Language Models (LLMs). We use the existing card-playing Schnapsen platform, built by researchers at the Vrije Universiteit Amsterdam. Card-playing games are well suited for reasoning benchmarking, since they are easy to implement, usually have simple rules, and include opportunities for strategic and forward thinking play.
In addition to evaluating LLM's ability to follow rules and play against a bot, the Schnapsen platform allows for pitting different models against each other.
Our research explores the strategic reasoning capabilities of various recent LLMs developed by OpenAI, Meta, Mistral, Microsoft, and Cohere. We found a significant correlation between models' inference times and reasoning ability, as measured by performance in the Schnapsen game. Our findings also indicate that specific smaller models outperform their larger competitors, even at lower inference times.
Given that inference times are correlated with the number of parameters, these models are delivering higher than expected performance given their weights. We believe that our findings highlight the potential of Schnapsen as a benchmark for evaluating LLMs' reasoning and planning skills in an adversarial and multi-agent environment.

Keywords: Large Language Model · Strategic Reasoning · Benchmarking · Game Playing.

1 Introduction

1.1 Card-playing ability as a benchmark

In recent years, Large Language Models (LLMs) have been shown to excel at text-centric tasks. Beyond natural language processing, researchers have recorded the potential of LLMs to perform well at multi-step reasoning and planning, including inductive reasoning and strategic decision-making in constrained environments. Although most of these models are intended for natural language processing, we are more interested in their multistep reasoning and planning capabilities in game-playing contexts.

The AI research and development has a long history of using games to test intelligence of programmed agents: Deep Blue beating Kasparov at chess, Watson AI winning Jeopardy, and, most recently, AlphaGo beating the best human Go player. [1] However, just showing a capacity to play a game is that difficult. As models become more advanced and capable, researchers find more intricate ways to test and benchmark their reasoning abilities.

Card games, in particular, offer a valuable testbed for verifying an agent's ability to interpret rules and reason about (partially) hidden information, while adapting optimal strategies.[2]

In this paper, we propose a hybrid system that uses LLMs to play a two-player, trick -taking card game known as Schnapsen; its two-phase structure - first a phase with imperfect information, then a more restricting phase with perfect information - and its relatively medium-scale state space offer an interesting opportunity for this purpose. The presence of special moves, such as Marriage and TrumpExchange, and the shift between phases create a rich environment for testing an AI agent's decision-making abilities.

1.2 Goals

Specifically, investigate the following aspects of LLM performance, based on similar papers[4]:

- How well an LLM-based bot can adhere to the game rules and if there are significant differences between each LLM. Measured in terms of returning valid moves from a list of possible moves/cards on hand.
- The strategic effectiveness of the LLM's chosen moves, benchmarked against both rule-based bots and other LLMs. Measured in terms of winning rate and average points won.
- Determine whether there are statistically significant differences in the performance of the models.

The results of these investigations are detailed in Section 4.
We utilize the framework outlined below:

- A game engine that defines Schnapsen's rules and phases.
- An LLM-based bot (referred to as LLMBot) instructed to pick the best next move via carefully constructed prompts containing the valid moves and the partial state of the game.
- A logging system that stores game data including moves, player perspectives, scores and special move declarations.

You can find detailed explanations on each part of the testing framework and set up of the experiments in Section 3.

We have integrated this system with baseline bots that each represent a distinct style of play. This includes an entirely random bot (RandBot) and a search-based Monte Carlo tree search bot (RdeepBot). Additionally, we conducted model-vs-model matches using gpt-4o-mini as an opponent. Through a series of experiments, we measure metrics such as inference completion time, win rate, invalid move percentage, use of special moves, and points scored to compare the performance of LLM agents utilizing different foundational models.

The list of LLMs used and tested in experiments, as per the deployment name mentioned in the Microsoft Azure AI platform:

- gpt-4,
- gpt-4o,
- gpt-4o-mini,
- o1-mini,
- o1-preview,
- Llama-3.3-70B-Instruct,
- Meta-Llama-3.1-405B-Instruct,

- Mistral-Large-2411,
- Mistral-small,
- Ministral-3B,
- Phi-4,
- Cohere-command-r-plus-08-2024.

2 Background

2.1 The game of Schnapsen

Schnapsen is a two-player trick-taking card game originating from Austria, closely related to Sixty-Six. It uses a 20-card deck consisting of the ranks Ace, Ten, King, Queen, and Jack across the four standard suits (clubs, diamonds, hearts, and spades). Each round (or deal) begins with five cards for each player, plus a face-up trump card on the bottom of the remaining talon (stock). The trump suit outranks all other suits during trick-taking. The objective for each deal is for a player to reach 66 or more trick points before the opponent, with between 1 and 3 game points awarded depending on the margin of victory.

A distinctive feature of the game is its two-phase structure: during Phase One (open stock), after each trick, the winner draws the top card of the talon, and the loser draws the next, preserving five cards in each player's hand. In this phase, no follow-suit requirement exists: the second player may play any card in response. During this phase, trump exchange (swapping the jack of trump for the face-up trump card) and marriages (declaring a king-queen pair) are possible. During Phase Two (closed or exhausted stock), players theoretically have perfect information. The follow-suit requirement now strictly applies, forcing the second player to play a card of the same suit or a trump.

Scoring goes as follows:

- The trick points for each card are: Ace = 11, Ten = 10, King = 4, Queen = 3, Jack = 2.
- A marriage in trump yields 40 immediate points, while a marriage in a non-trump suit yields 20 points.
- The player who reaches 66 deal points first wins. The winner scores 1-3 game points for that deal.
- Game points depend on the score (taken tricks) of the opponent: 3 points if the opponent won no tricks, 2 points if the opponent won tricks with total value less than 33, and 1 point otherwise.

2.2 LLMs for Strategic Game Play

Large Language Models (LLMs) are primarily used for text-based tasks, but emerging studies [link references] show that, when properly prompted, they can exhibit basic planning and strategic thinking skills. The novel challenge we propose here is real-time move generation under explicit rules, partial and full knowledge, and an adversarial setting. We aim to see whether a specialized prompt containing the Schnapsen state representation and the valid moves, along with a strict output format, can elicit high-quality moves from an LLM.

Potential challenges include:

- Rule-following: LLMs might produce outputs that are not in the set of valid moves (invalid move type, wrong suit, etc.).
- Long-term planning: If the LLM does not fully understand how the game state is represented, it might not handle complex multi-trick planning well.
- Strategic depth: Even if the LLM can parse the partial state of the game, it may rely on shallow heuristics or minimal strategic reasoning, and return moves at random or the first available move in the list.
- Structured output: LLMs differ in their ability to follow string output format instructions.

To solve the first challenge, rule-following, we check whether the suggested move is a valid move. If the returned move is not valid, we prompt LLM with a new message, indicating that the move is invalid, and ask for a new suggestion. In case repeated invalid moves are returned, after the 10th invalid move we simply return a random move from available moves.

We hope that high performance in games against different types of opponents would indicate some reasoning and planning ability. Win rates are provided in Section 4.

To overcome the structured output challenge, we decided not to test the models on their ability to produce a valid JSON. Instead, we prompt for a natural language description of the move, and then make an ancillary call to another llm (gpt-4o-mini) to transform natural language description of a move into a JSON representation of the move as python class. In this manner, we do not depend on a particular model's ability to return structured output, and we can test smaller and older models. The 'gpt-4o-mini' model was chosen for this task based on high inference speed, expected quality and cost.

3 Experiments

This section details the experimental setup and our approach in pitting various LLMs against baseline programmed bots (RandBot and RdeepBot) and an LLM-based bot ('gpt-4o-mini' model) in the Schnapsen Game Engine.

3.1 Experimental Setup

Our experiments involved playing a large number of Schnapsen matches using the Schnapsen Game Engine found within this repository: Schnapsen Game Engine[4]. This engine implements the complete rules of Schnapsen, including both phases of play, special moves (marriage and trump exchange), and scoring. We integrated several LLMs as automated players (LLMBots), prompting them with the following:

- Current game state.
- Data from previous tricks in the same game.
- A "cleaned" version of the Schnapsen rules, originating from Schnapsen Rules[5]. We edited it to remove unnecessary information and to reduce prompt size and input tokens.
- Strategic guidelines as outlined in Schnapsen Strategy[6].
- The list of valid moves for that trick.

[4] https://github.com/intelligent-systems-course/schnapsen.git
[5] http://psellos.com/schnapsen/rules.html
[6] http://psellos.com/schnapsen/strategy.html

The code used is publicly available on GitHub platform.[7] The prompt, which is available in bot.py file[8], lines 87-124, was the result of iterative design and continuous evaluation to ensure that we provide all necessary information in a structured format, minimizing ambiguity and maximizing the likelihood of receiving a valid move in return. For this purpose, we also kept track of how many invalid moves the LLM returned. In the early stages of our research, we thought about trying different prompts and evaluating their effectiveness. However, we decided that first we would like to document the general ability of an LLM to play Schnapsen and test the Schnapsen python platform. Given the eventual feedback to the present paper, additional research can be done on prompt engineering/effectiveness. Once we found a prompt that worked consistently, we have used that same prompt in all experiments.

For comparison, we employed two baseline-programmed bots, RandBot and RdeepBot, as well as an LLM-based bot utilizing the gpt-4o-mini model.

RandBot selects a legal move entirely at random. It serves as a control, establishing a baseline performance level and verifying that the LLMBots can outperform a purely random strategy.

RdeepBot utilizes Monte Carlo Tree Search (MCTS) to select its moves. The MCTS decision-making algorithm explores possible future game states through repeated simulations, allowing the bot to make informed decisions based on probabilistic outcomes. RdeepBot represents a strong opponent, providing a challenging benchmark for LLMBots. This choice is particularly relevant because it allows us to evaluate how well LLMs can compete with a bot that explicitly simulates future game states, thus testing their own implicit planning abilities. A working version of RdeepBot is already present in the main Schnapsen repository of this course, and it takes two parameters: depth of search and sample size.[9]

Matches were conducted with each LLMBot playing against both RandBot and RdeepBot. We recorded various game metrics for each match, including:

- **Leader/Follower**: the Player Perspective of the LLM at the beginning of each match.
- **Model**: The specific LLM used.
- **Won**: A boolean indicating whether the LLM won the match or not.
- **Score/Opponent Score**: The final scores for both players.
- **Turns**: The total number of turns played in the match.
- **Invalid Moves**: The number of illegal moves attempted by the LLMBot.
- **Average Completion Time**: The average inference time (calculated as the average of moves in one match) taken by the LLMBot to generate a move.

Intermediate results for turns were stored in a NoSQL database, and final match results were written to a cloud-hosted PostgreSQL database instance. The table then was accessed for analysis using Pandas DataFrames in a Jupyter Notebook environment.

3.2 Notes on Data Analysis and Metrics

We then processed the data for analysis by computing several key metrics:

- **Win Ratio**: The percentage of matches won by the LLMBot over total matches played.

[7] https://github.com/storks-amsterdam/vu-ai-pris
[8] https://github.com/storks-amsterdam/vu-ai-pris/blob/main/schnapsen_llm_bench/bot.py
[9] We used a depth of 5 with 10 samples.

- **Average Score**: The average match points scored by the LLMBot.
- **Average Game Points**: The average game points earned by the LLMBot at the end of each match.
- **Average Completion Time**: The average time it took the LLM to return a Move.
- **Invalid Moves**: The number of Invalid Moves returned by the LLM, along with the total matches and ratio of matches with Invalid Moves.

4 Analysis of Results

4.1 Overall performance

The following section presents a comparison of the LLMs against the two baseline bots described above: RandBot, which plays moves randomly, and RdeepBot, a Monte Carlo Tree Search based bot. Additionally, we pitted the LLMs against gpt-4o-mini, to test matches played by an LLMBot on both sides. These baselines represent different ends of strategic play, allowing a performance assessment of the LLMs relative to both a random approach and a more sophisticated and planned strategy.

Table 4.1 presents an overview of the matches played by each LLM against three different opponents. The total number of matches used for analysis is 10539.

Table 1. Number of matches per model and opponent

Model	gpt-4o-mini	randbot	rdeepbot	Total
Cohere-command-r-plus-08-2024	133	299	571	1003
Llama-3.3-70B-Instruct	121	497	506	1124
Meta-Llama-3.1-405B-Instruct	109	809	140	1058
Ministral-3B	113	597	333	1043
Mistral-Large-2411	118	441	440	999
Mistral-small	107	534	461	1102
Phi-4	93	479	467	1039
gpt-4	111	527	452	1090
gpt-4o	107	368	595	1070
gpt-4o-mini	nan	250	612	862
o1-mini	1	123	6	130
o1-preview	nan	18	1	19
Total	1013	4942	4584	10539

For our main analysis of LLMs' performance we chose to concentrate on metrics for wins, score of the tricks taken, final winning points, ratio of matches with invalid moves, and inference completion time. In table 4.1 we present an overview of aggregated metrics for all models playing against RandBot. The same metrics are presented for RdeepBot and gpt-4o-mini in Appendix A, tables 4 and 5.[10]

[10] Due to different inference method used by o1 models, we decided not to use them in our analysis.

Table 2. Performance of LLMs Against Baseline RandBot

Model	Win Ratio (%)	Average Score	Average Points	Average Completion Time	IMR (%)
Cohere-command-r-plus-08-2024	60.87	53.8829	1.06689	1599.47	3.68
Llama-3.3-70B-Instruct	67.81	62.827	1.30986	870.419	6.04
Meta-Llama-3.1-405B-Instruct	68.5	62.3175	1.37375	2113.34	3.75
Ministral-3B	45.39	48.4422	0.80402	455.491	39.7
Mistral-Large-2411	63.72	60.2449	1.22449	2167.9	15.19
Mistral-small	63.11	57.6479	1.2191	992.232	36.14
Phi-4	62.84	58.8977	1.119	2641.35	33.82
gpt-4	68.69	63.2732	1.37571	1688.9	3.98
gpt-4o	76.63	65.0598	1.52989	2443.71	3.53
gpt-4o-mini	59.2	56.548	1.1	475.682	13.2
o1-mini	79.34	69.1653	1.61983	15054.6	7.44
o1-preview	90.91	71.6364	2.09091	26189.4	36.36

There is some variation in win ratios across the LLMs we tested against the RandBot, suggesting varying levels of strategic competence and understanding of the game. The best performing LLMs achieve win ratios close to or above 70%, while the underperforming ones achieve less than 50%. However, most models have a winning ratio between 60 and 70%. This shows stability in performance and general ability to understand and play the game. Invalid Move Ratios (IMRs) also vary significantly, with some models exhibiting IMRs as high as 30/40%, while others, like Meta's Llama models and the gpt-4 series models, show among the lowest registered IMRs. We did not find correlation between IMR's and winning performance, but we also did not test it statistically.

4.2 Testing Models' Performance

To test how different models perform compared to each other, we decided to use chat inference completion time as a proxy for model's parameter size (weights) and inference costs. Since the number of tokens sent to each mode is approximately the same (the length of text is definitely the same), and the compute infrastructure that runs these models is also the same (we are using dedicated Azure AI model deployments with dedicated capacity) - the difference in inference completion time should correlate strongly with number of parameters.

We can also see that completion times by model vary between them, but stable for each model. See Figure 1 for completion times by model.

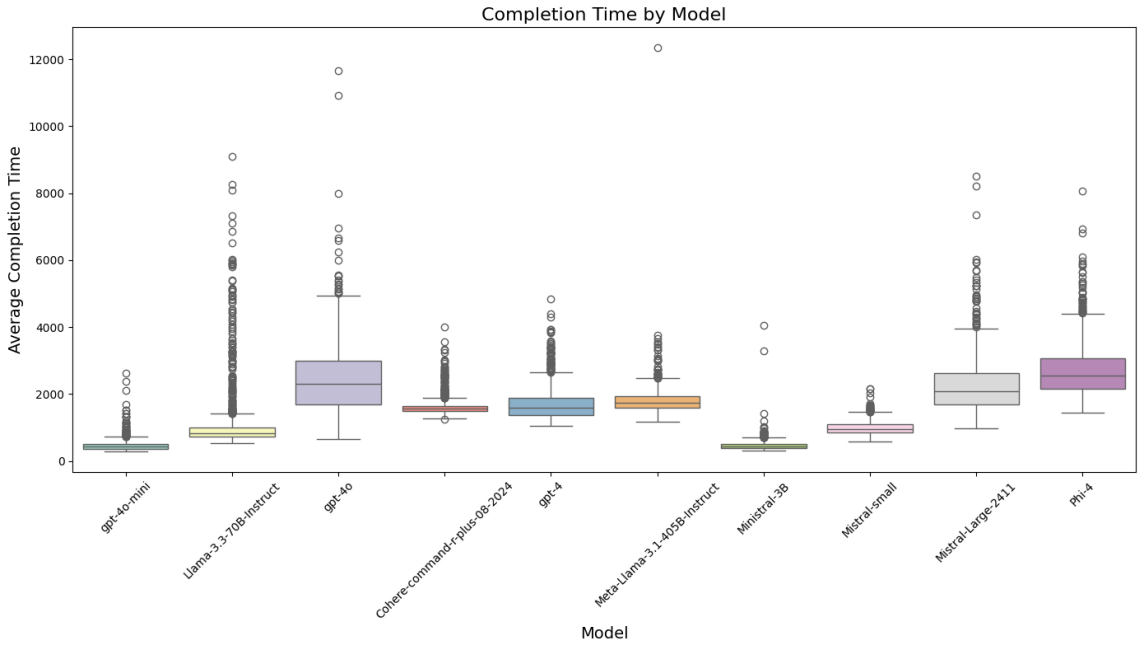
The plot of the average points achieved versus the completion time confirmed our expectations. See Figure 2.[11]

Figure 8 shows that, generally, points earned tend to increase with average completion time. However, the flatter fit line for RdeepBot indicates that a longer completion time does not necessarily imply higher points when playing against a stronger opponent.

We ran an OLS regression analysis, with attained points as a dependent variable, and completion time and opponents (one-hot encoded) as independent variables. See the OLS output in Figure 4.

[11] There is also a scatter for Win Ratio against Completion Time in the appendix Table **??**

Fig. 1. Completion Time per Model



The regression model reveals a statistically significant relationship between average completion time and points earned ($p < 0.001$). The positive coefficient for `avg_completion_time` (0.1047) associates a longer completion time with higher points, but a low R-squared value of 0.193 shows that this explains only part of the variation in points earned.

As expected, the opponent impacts the score significantly ($p < 0.001$), and confirms that playing against RdeepBot (negative `opponent_randbot` coefficient = -0.5773) leads to lower scores than playing against RandBot (positive `opponent_randbot` coefficient = 0.3396). This is compared to gpt-4o-mini as a baseline opponent, since it was dropped in regression when creating one-hot encoding.

Now we know that completion time and opponent correlate significantly with the performance of the model. Due to our assumptions made earlier about the relationship between completion time and model size, our findings make sense. Bigger models, with a higher number of parameters, would perform better against a baseline bot. We also see that the type of Bot is statistically significant in the performance of the model. With this, we have shown that models can play a game of Schnapsen, that the parameter/model size plays a role, and the ability of an opponent plays a role.

Next we would like to show that the choice of model makes a difference: that, keeping everything else equal, some models perform better, even taking into account their size and opponent's strengths.

For that, we run another OLS regression analysis, now also including models (one-hot encoded) as independent variables. The regression tests whether there the model choice has a non-zero effect on the performance of the LLMBot, and if this effect is negative, or positive, keeping all else (completion time and opponent type) equal. The output of the OLS regression analysis is presented

Fig. 2. Avg. Points vs Avg. Completion Time (wins only)

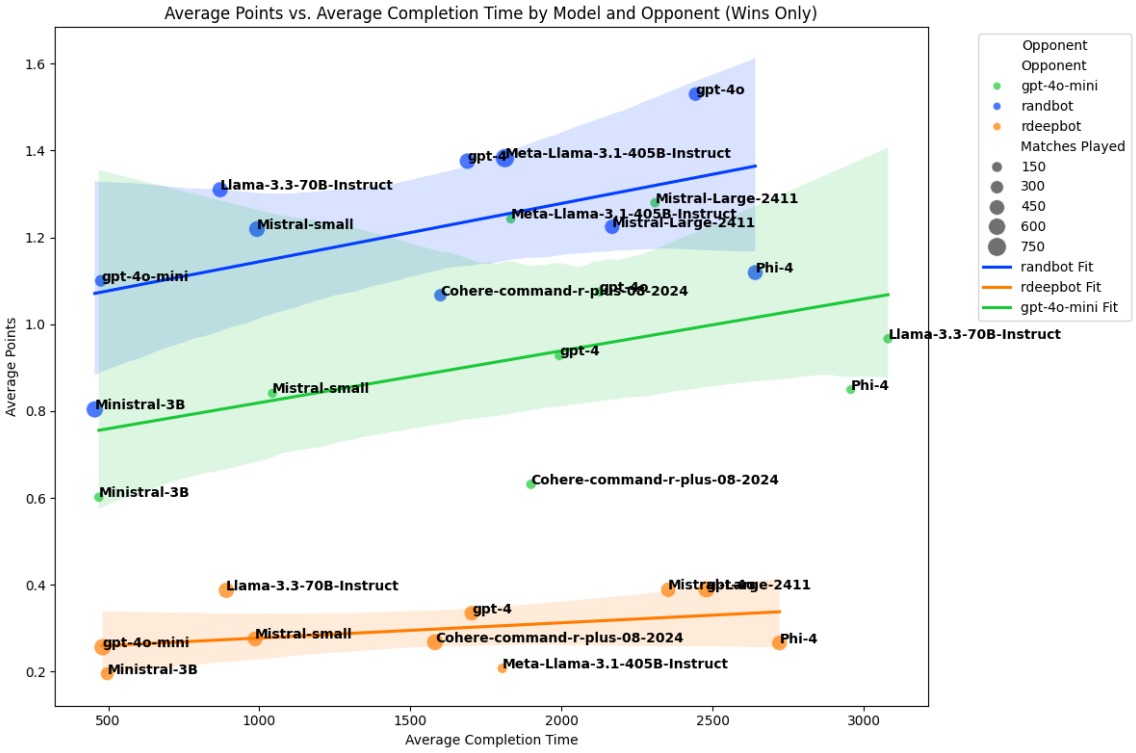


Fig. 3. OLS Regression Analysis, Completion Time, without Models

OLS Regression Results

Dep. Variable:	points	R-squared:	0.193
Model:	OLS	Adj. R-squared:	0.193
Method:	Least Squares	F-statistic:	800.5
Date:	Tue, 28 Jan 2025	Prob (F-statistic):	0.00
Time:	22:37:45	Log-Likelihood:	-13429.
No. Observations:	10028	AIC:	2.687e+04
Df Residuals:	10024	BIC:	2.689e+04
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.7239	0.034	21.536	0.000	0.658	0.790
avg_completion_time	0.1047	0.009	12.248	0.000	0.088	0.121
opponent_randbot	0.3396	0.032	10.474	0.000	0.276	0.403
opponent_rdeepbot	-0.5773	0.032	-17.817	0.000	-0.641	-0.514

Omnibus:	687.639	Durbin-Watson:	1.959
Prob(Omnibus):	0.000	Jarque-Bera (JB):	830.444
Skew:	0.700	Prob(JB):	4.69e-181
Kurtosis:	2.827	Cond. No.	12.6

in Figure 4. The interesting statistic is the level of significance and the sign of the coefficient. We see that almost all models have non-zero effect on the outcome, at close to 99.99% significance level.

Higher coefficients signify better performance vis-a-vis models present in the analysis.

Fig. 4. OLS Regression Analysis, Completion Time, with Models

Dep. Variable:	points	R-squared:	0.207
Model:	OLS	Adj. R-squared:	0.206
Method:	Least Squares	F-statistic:	200.6
Date:	Tue, 28 Jan 2025	Prob (F-statistic):	0.00
Time:	22:59:14	Log-Likelihood:	-13345.
No. Observations:	10028	AIC:	2.672e+04
Df Residuals:	10014	BIC:	2.682e+04
Df Model:	13		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.6243	0.048	12.964	0.000	0.530	0.719
avg_completion_time	9.793e-05	1.44e-05	6.800	0.000	6.97e-05	0.000
opponent_randbot	0.3334	0.033	10.245	0.000	0.270	0.397
opponent_rdeepbot	-0.5778	0.033	-17.645	0.000	-0.642	-0.514
model_Llama-3.3-70B-Instruct	0.2367	0.041	5.838	0.000	0.157	0.316
model_Meta-Llama-3.1-405B-Instruct	0.2649	0.045	5.864	0.000	0.176	0.353
model_Ministral-3B	-0.0890	0.044	-2.021	0.043	-0.175	-0.003
model_Mistral-Large-2411	0.1497	0.043	3.491	0.000	0.066	0.234
model_Mistral-small	0.1461	0.041	3.553	0.000	0.065	0.227
model_Phi-4	-0.0718	0.044	-1.646	0.100	-0.157	0.014
model_gpt-4	0.1836	0.040	4.557	0.000	0.105	0.263
model_gpt-4o	0.1946	0.042	4.637	0.000	0.112	0.277
model_gpt-4o-mini	0.1434	0.046	3.140	0.002	0.054	0.233
model_o1-mini	0.3336	0.244	1.370	0.171	-0.144	0.811

Omnibus:	656.433	Durbin-Watson:	1.974
Prob(Omnibus):	0.000	Jarque-Bera (JB):	786.713
Skew:	0.681	Prob(JB):	1.47e-171
Kurtosis:	2.837	Cond. No.	5.14e+04

4.3 Error Analysis and Rule Compliance

Analyzing rule violations provides insight into the LLMs' understanding of game rules and mechanics, and strategy. Figure 3 shows the ratio of invalid moves by model and opponent.

This histogram reveals strong differences in rule compliance across LLMs; a wide range of IMRs was observed, with models such as Ministral-3B, Mistral-small, and Phi-4 exhibiting high rates (over 30-40%), which may indicate poor understanding of game rules or the implementation of a weak strategy. However, some models, such as o1-mini, gpt-4, and Meta-Llama-3.1-405B-Instruct, demonstrated much lower IMRs (below 10%).

The choice of opponent does not seem to influence the IMR as much as the choice of the model (visually).

5 Conclusion

5.1 Discussion

The findings of this study demonstrate the potential of Schnapsen as a benchmarking platform to evaluate the strategic reasoning and planning abilities of Large Language Models (LLMs). The use

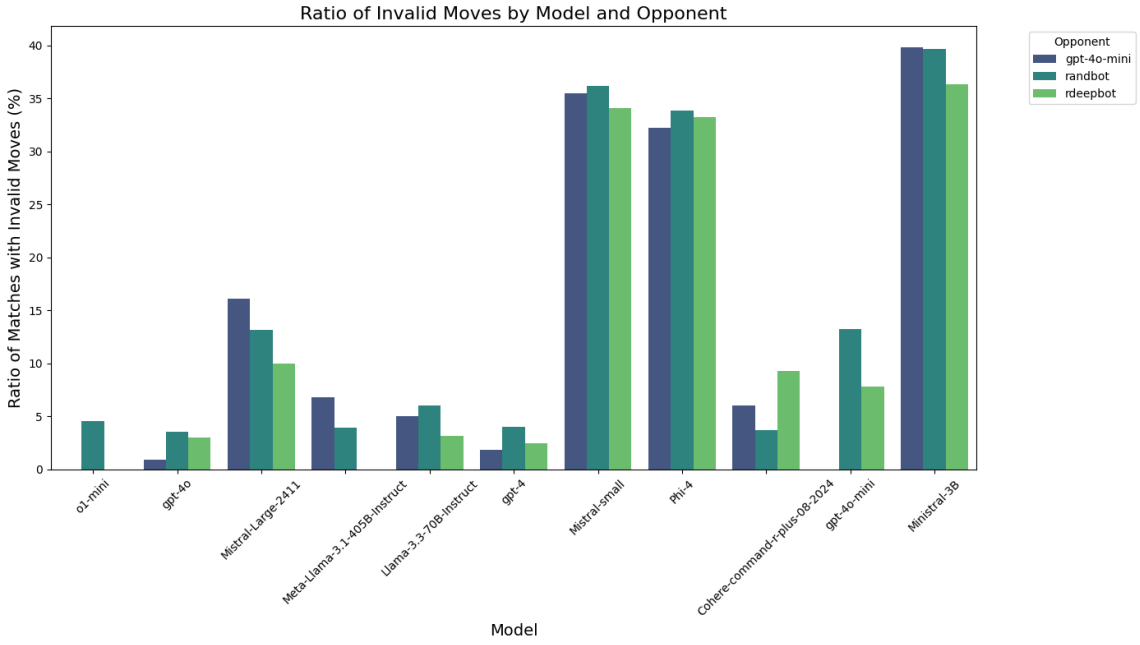


Fig. 5. IMR by Model and Opponent

of Schnapsen as a benchmarking tool opens up new avenues for evaluating LLMs in structured, adversarial, and multi-agent environments.

By pitting LLM-based bots against both baseline bots (RandBot and RdeepBot) and other LLMs, we observed significant differences in performance metrics such as win ratio, average points scored, and invalid move ratios (IMRs). The results also revealed a strong correlation between inference completion time—used as a proxy for model size—and performance, underscoring the role of model complexity in strategic reasoning tasks.

Despite the general assumption that larger models and more parameters are always better, some models show an edge in performance against similarly sized models, suggesting that factors other than just parameter count, such as architecture optimization or fine-tuning, may play a role. Additionally, the ability of most LLMs to consistently outperform RandBot indicates a baseline understanding of the game's rules and strategy. Lower and less varying levels of success against RdeepBot might imply a ceiling level in planning and reasoning abilities of current LLMs.

5.2 Future Improvements

While this study provides a comprehensive evaluation of LLMs using Schnapsen, there are several avenues for improvement and extension:

Prompt Engineering and Optimization The current study employed a single iterative prompt design for all LLMs. Future research could explore the impact of varying prompt structures and strategies on performance. For example, prompts could be tailored to individual models based on

their architecture or fine-tuning history. Additionally, techniques such as chain-of-thought prompting could be employed to encourage multi-step reasoning.

Broader Model Selection The selection of LLMs in this study was limited to recent models from OpenAI, Meta, Mistral, Microsoft, and Cohere. Future studies could include other prominent models, including fine-tuned or domain-specific LLMs, to provide a more comprehensive evaluation. Additionally, open-source models with customizable architectures could be adapted specifically for strategic reasoning tasks.

References

1. David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. "Mastering the game of Go with deep neural networks and tree search." *Nature*, vol. 529, no. 7587, pp. 484–489, 2016. [Online]. Available: https://www.nature.com/articles/nature16961
2. Davide Paglieri, Bartłomiej Cupiał, Samuel Coward, Ulyana Piterbarg, Maciej Wolczyk, Akbir Khan, Eduardo Pignatelli, Łukasz Kuciński, Lerrel Pinto, Rob Fergus, Jakob Nicolaus Foerster, Jack Parker-Holder, Tim Rocktäschel. "BALROG: Benchmarking Agentic LLM and VLM Reasoning On Games." arXiv preprint arXiv:2411.13543, 2024. [Online]. Available: https://arxiv.org/abs/2411.13543
3. Georgios N. Yannakakis, Julian Togelius. "Game-Playing as a Testbed for General Artificial Intelligence." In *Artificial Intelligence and Games*, pp. 409–423, Springer, 2018. [Online]. Available: https://www.academia.edu/47849987/Georgios_N_Yannakakis_and_Julian_Togelius_Artificial_Intelligence_and_Games
4. Bahar Bateni, Jim Whitehead. "Language-Driven Play: Large Language Models as Game-Playing Agents in Slay the Spire." In *Proceedings of the 19th International Conference on the Foundations of Digital Games*, 2024, pp. 20. Association for Computing Machinery, New York, NY, USA. [Online]. Available: https://doi.org/10.1145/3649921.3650013
5. Sebastien Bubeck, Varun Chandrasekaran, Ronan Collobert, Karl Cobbe, Trevor Darrell, Jacob Hilton, et al. "Sparks of Artificial General Intelligence: Early experiments with GPT-4." arXiv preprint arXiv:2303.12712, 2023. [Online]. Available: https://arxiv.org/abs/2303.12712
6. Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, et al. "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models." In *Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html
7. Philipp Mondorf, Barbara Plank. "Beyond Accuracy: Evaluating the Reasoning Behavior of Large Language Models – A Survey." arXiv preprint arXiv:2404.01869, 2024. [Online]. Available:

A Detailed Summary Statistics

Table 3. Detailed Model Performance Metrics

Model	Opponent	Matches	Won	Win (%)	Score	Points	Avg. CT (ms)	IMR (%)
Cohere-command-r-plus-08-2024	gpt-4o-mini	133	55	41.35	44.6992	0.631579	1899.42	6.02
Cohere-command-r-plus-08-2024	randbot	299	182	60.87	53.8829	1.06689	1599.47	3.68
Cohere-command-r-plus-08-2024	rdeepbot	571	115	20.14	26.8704	0.267951	1581.74	9.28
Llama-3.3-70B-Instruct	gpt-4o-mini	121	63	52.07	52.4132	0.958678	3238.27	4.96
Llama-3.3-70B-Instruct	randbot	497	337	67.81	62.827	1.30986	870.419	6.04
Llama-3.3-70B-Instruct	rdeepbot	506	136	26.88	37.3794	0.387352	890.943	3.16
Meta-Llama-3.1-405B-Instruct	gpt-4o-mini	106	71	66.98	57.5472	1.21698	2268.81	6.6
Meta-Llama-3.1-405B-Instruct	randbot	800	548	68.5	62.3175	1.37375	2113.34	3.75
Meta-Llama-3.1-405B-Instruct	rdeepbot	135	21	15.56	31.0667	0.207407	2457.34	0.74
Ministral-3B	gpt-4o-mini	113	42	37.17	43.6372	0.60177	468.748	39.82
Ministral-3B	randbot	597	271	45.39	48.4422	0.80402	455.491	39.7
Ministral-3B	rdeepbot	333	56	16.82	25.8018	0.195195	496.662	36.34
Mistral-Large-2411	gpt-4o-mini	118	76	64.41	58.2966	1.27966	2309.6	16.1
Mistral-Large-2411	randbot	441	281	63.72	60.2449	1.22449	2167.9	15.19
Mistral-Large-2411	rdeepbot	440	113	25.68	36.1364	0.388636	2353.42	10
Mistral-small	gpt-4o-mini	107	55	51.4	49.5234	0.841121	1043.38	35.51
Mistral-small	randbot	534	337	63.11	57.6479	1.2191	992.232	36.14
Mistral-small	rdeepbot	461	91	19.74	29.9848	0.275488	985.978	34.06
Phi-4	gpt-4o-mini	93	46	49.46	51.7204	0.849462	2957.75	32.26
Phi-4	randbot	479	301	62.84	58.8977	1.119	2641.35	33.82
Phi-4	rdeepbot	466	96	20.6	30.4034	0.266094	2721.65	33.26
gpt-4	gpt-4o-mini	111	57	51.35	53.8559	0.927928	1992.74	1.8
gpt-4	randbot	527	362	68.69	63.2732	1.37571	1688.9	3.98
gpt-4	rdeepbot	451	109	24.17	36.3149	0.334812	1703.24	2.44
gpt-4o	gpt-4o-mini	107	67	62.62	57.0374	1.07477	2126.33	0.93
gpt-4o	randbot	368	282	76.63	65.0598	1.52989	2443.71	3.53
gpt-4o	rdeepbot	595	162	27.23	37.4908	0.389916	2479.87	3.03
gpt-4o-mini	randbot	250	148	59.2	56.548	1.1	475.682	13.2
gpt-4o-mini	rdeepbot	612	119	19.44	31.0294	0.256536	481.542	7.84
o1-mini	gpt-4o-mini	1	1	100	80	2	25785	0
o1-mini	randbot	121	96	79.34	69.1653	1.61983	15054.6	7.44
o1-mini	rdeepbot	6	1	16.67	41.1667	0.166667	15628.3	0
o1-preview	randbot	11	10	90.91	71.6364	2.09091	26189.4	36.36

Table 4. Model Performance against RDeepBot

Model	Win Ratio (%)	Average Score	Average Points	Average Completion Time	IMR (%)
Cohere-command-r-plus-08-2024	20.14	26.8704	0.267951	1581.74	9.28
Llama-3.3-70B-Instruct	26.88	37.3794	0.387352	890.943	3.16
Meta-Llama-3.1-405B-Instruct	15.56	31.0667	0.207407	2457.34	0.74
Ministral-3B	16.82	25.8018	0.195195	496.662	36.34
Mistral-Large-2411	25.68	36.1364	0.388636	2353.42	10
Mistral-small	19.74	29.9848	0.275488	985.978	34.06
Phi-4	20.6	30.4034	0.266094	2721.65	33.26
gpt-4	24.17	36.3149	0.334812	1703.24	2.44
gpt-4o	27.23	37.4908	0.389916	2479.87	3.03
gpt-4o-mini	19.44	31.0294	0.256536	481.542	7.84
o1-mini	16.67	41.1667	0.166667	15628.3	0

Table 5. Model Performance against gpt-4o-mini

Model	Win Ratio (%)	Average Score	Average Points	Average Completion Time	IMR (%)
Cohere-command-r-plus-08-2024	20.14	26.8704	0.267951	1581.74	9.28
Llama-3.3-70B-Instruct	26.88	37.3794	0.387352	890.943	3.16
Meta-Llama-3.1-405B-Instruct	15.56	31.0667	0.207407	2457.34	0.74
Ministral-3B	16.82	25.8018	0.195195	496.662	36.34
Mistral-Large-2411	25.68	36.1364	0.388636	2353.42	10
Mistral-small	19.74	29.9848	0.275488	985.978	34.06
Phi-4	20.6	30.4034	0.266094	2721.65	33.26
gpt-4	24.17	36.3149	0.334812	1703.24	2.44
gpt-4o	27.23	37.4908	0.389916	2479.87	3.03
gpt-4o-mini	19.44	31.0294	0.256536	481.542	7.84
o1-mini	16.67	41.1667	0.166667	15628.3	0

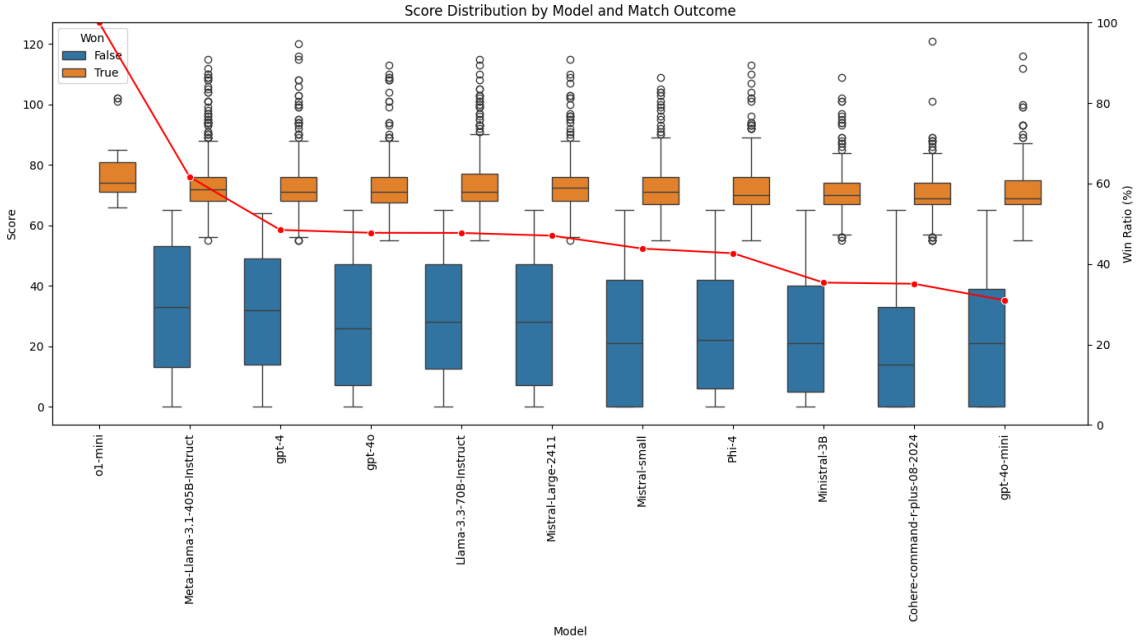


Fig. 6. Score Distribution by Model and Match Outcome

Fig. 7. Win Ratio vs Avg. Completion Time by Model and Opponent

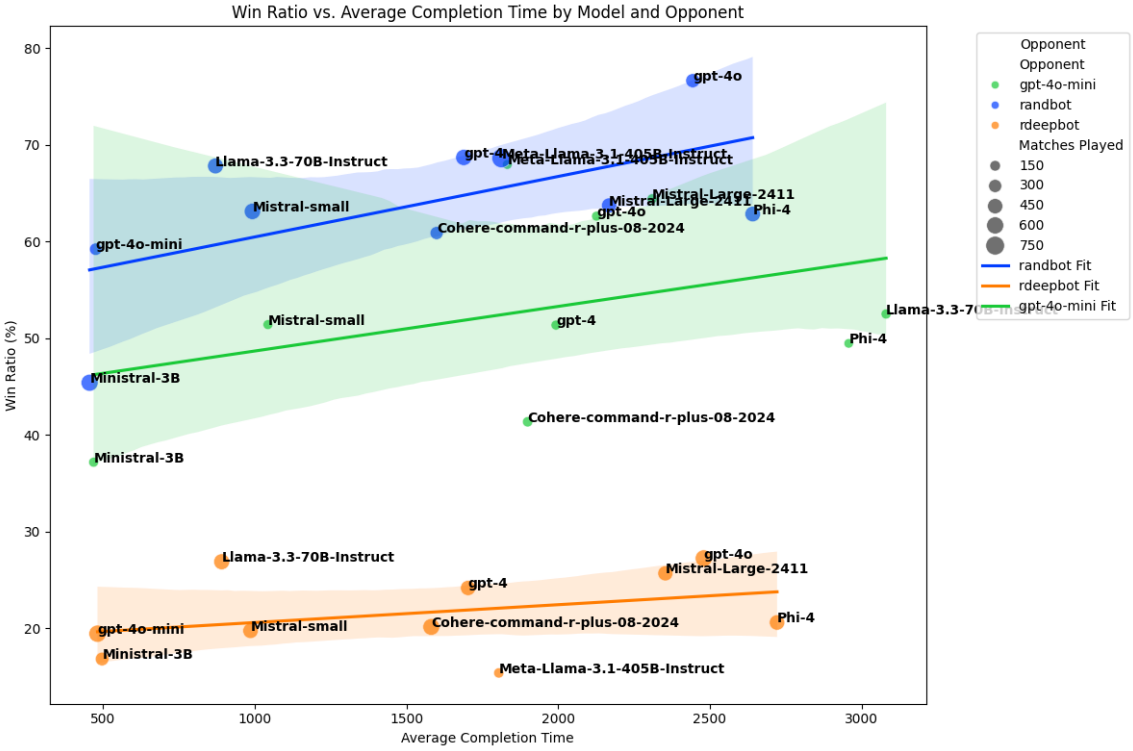


Fig. 8. Average Points vs Completion Time by Model and Opponent