

Usando LLMs para Programar Jogos de Tabuleiro e Variações

Álvaro Guglielmin Becker¹, Lana Bertoldo Rossato¹, Anderson Rocha Tavares¹

¹Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15064 – Porto Alegre – RS – Brasil

{agbecker, lbrossato, artavares}@inf.ufrgs.br

Abstract. *Creating programs to represent board games can be a time-consuming task. Large Language Models (LLMs) arise as appealing tools to expedite this process, given their capacity to efficiently generate code from simple contextual information. In this work, we propose a method to test how capable three LLMs (Claude, DeepSeek and ChatGPT) are at creating code for board games, as well as new variants of existing games.*

Resumo. *A criação de programas que representem jogos de tabuleiro pode ser uma tarefa demorada. Large Language Models (LLMs) surgem como ferramentas interessantes para acelerar esse processo, dada sua capacidade para geração eficaz de código a partir de informações contextuais simples. Neste trabalho, propomos um método para testar a capacidade de três LLMs (Claude, DeepSeek e ChatGPT) de criar código para jogos de tabuleiro, bem como de novas variantes de jogos existentes.*

1. Introdução

Jogos de tabuleiro são um assunto importante na área de Inteligência Artificial (IA). Muitos trabalhos exploraram a criação de algoritmos para jogar estes jogos [Genesereth et al. 2005, Nielsen et al. 2014], e outros a criação de novos jogos a partir de variações sobre jogos existentes [Browne 2011, Rossato et al. 2023, Todd et al. 2024]. Em ambos os casos, é necessária a representação dos jogos estudados em código, cuja implementação pode ser uma tarefa demorada e trabalhosa.

Nesse contexto, Large Language Models (LLMs) possuem grande potencial, das suas capacidades em geração de código a partir de descrições em linguagem natural [Coignon et al. 2024]. Além disso, devido à massiva quantidade de dados usada nos seus treinamentos, terão familiaridade com as regras e implementações existentes dos jogos de tabuleiro mais populares. Assim, seu uso poderia representar um ganho substancial de eficiência na implementação não só de jogos existentes, de variantes desses jogos, ou até mesmo de novos jogos descritos por analogia aos já existentes.

Neste trabalho, propomos um método para avaliar o desempenho de três LLMs (Claude 3.7 Sonnet, DeepSeekV3 e ChatGPT-4o) na tarefa de geração de código em Python para seis jogos de tabuleiro comuns, conhecidos pelos modelos. Testamos a implementação tanto dos jogos como de duas variações de cada jogo, modificando seu equipamento (tabuleiro, peças) e suas regras, para verificar a capacidade de raciocínio em alto nível dos modelos sobre aspectos dos jogos. Adicionalmente, testamos o efeito do uso da Boardwalk API [Becker et al. 2025], uma plataforma própria para desenvolvimento de jogos de tabuleiro em Python que permite fácil integração com agentes de IA jogadores.

Esperamos obter uma taxa alta de sucesso, sobretudo nas implementações dos jogos sem modificações e sem uso da API, visto que os modelos estarão essencialmente reproduzindo implementações que terão certamente encontrado em seu treinamento. Havendo boa performance na implementação dos jogos com variações de regras, a descrição de jogos novos ou pouco conhecidos em termos de jogos conhecidos consolidaria-se como estratégia útil para designers de jogos e pesquisadores.

2. Trabalhos Relacionados

A maioria dos trabalhos existentes na geração de código para jogos de tabuleiro com LLMs não utiliza linguagens de programação de propósito geral, mas sim Game Description Languages (GDLs), linguagens de descrição de jogos com gramática própria, como a Ludii GDL [Piette et al. 2020]. Todd et al. (2024) fizeram fine-tuning em uma LLM para completar descrições parciais de jogos na Ludii GDL, e Tanaka e Simo-Serra (2024) usaram uma LLM para gerar descrições em Ludii a partir de prompts em linguagem natural. Entretanto, ambos os trabalhos relataram alta frequência de erros nas implementações devido à complexidade da gramática Ludii e seu relativo desconhecimento por parte dos modelos.

Em [Becker et al. 2025], usamos descrições em linguagem natural de regras de jogos como prompts para geração de código em Python por LLMs, obtendo uma taxa de sucesso razoável. Além disso, introduzimos a Boardwalk API, que permite uma relativa padronização do formato do código e fácil integração com agentes de IA jogadores.

A abordagem proposta neste trabalho é nova ao se utilizar explicitamente dos conhecimentos prévios dos modelos quanto às regras dos jogos, invocando-os apenas através dos nomes dos jogos. Além disso, testamos o raciocínio em alto nível dos modelos sobre este conhecimento, conciliando o que sabem sobre as regras originais com as variações pedidas, ao invés de descrever as regras já com as modificações incorporadas.

3. Metodologia Proposta

Serão realizados testes com três LLMs: Claude Sonnet 3.7 [Anthropic 2024], DeepSeekV3 [Liu et al. 2025] e ChatGPT-4o [OpenAI et al. 2024]. Embora haja versões mais atualizadas destes modelos, são as mesmas versões usadas em [Becker et al. 2025], permitindo assim uma comparação justa de desempenho. Os modelos serão acessados através da plataforma Poe¹.

Foram selecionados seis jogos de tabuleiro para teste: Jogo da Velha, Resta Um, Reversi, Moinho, Damas e Xadrez. Por serem jogos consagrados, espera-se que as LLMs tenham conhecimento suficiente sobre as regras dos jogos e implementações existentes deles em código². Para cada jogo, serão pedidas três variações: uma original, uma mudando o equipamento (formato do tabuleiro ou distribuição de peças) e uma mudando as regras em si (condições de vitória ou funcionamento de peças).

A cada modelo, para cada jogo, serão pedidas duas formas de implementação em Python: uma com a Boardwalk API, e outra sem (implementação independente).

¹<https://poe.com>

²Como verificação primária desta hipótese, pedimos a cada modelo que descrevesse as regras de cada jogo, e todas foram fornecidas corretamente.

Nos testes com a API, será fornecida apenas a documentação da API, e não seu código fonte. Da mesma forma que foi feito em [Becker et al. 2025], isso permitiria verificar se o uso da API atrapalha na geração de código, embora enriqueça o resultado final devido à padronização do código e interface para jogadores humanos e agentes de IA. No total, serão executados 108 testes (3 LLMs, 6 jogos, 3 variações, 2 formas de implementação).

Serão usados prompts padronizados em todos os testes. Estes diferirão dependendo do tipo de teste (independente ou com API, e jogo original ou variação); entretanto, testes na mesma categoria usarão o mesmo prompt, mudando apenas o nome do jogo e a mudança nas regras, se aplicável. A Figura 1 mostra um exemplo de prompt para variações de regra.

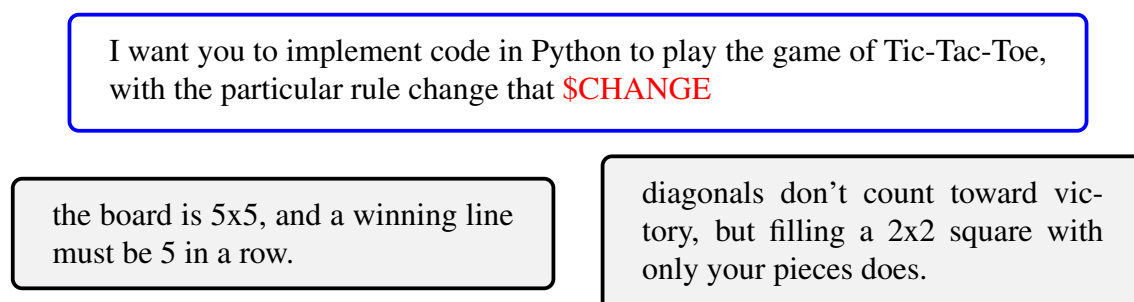


Figura 1. Prompt usado para a implementação independente do Jogo da Velha com variação de regras. A expressão “\$CHANGE” é substituída pela descrição da regra modificada nos retângulos cinza.

Os códigos gerados serão avaliados por *playtest* manual, executando o programa e jogando partidas representativas do jogo para verificar a implementação. Os resultados serão quantificados pela ocorrência de erros: de sintaxe de Python, no uso da API, na movimentação das peças, nas condições de término e vitória, nos efeitos das jogadas (e.g. capturas e promoções), na formatação do tabuleiro, e na ordem de ação dos jogadores. A ocorrência de erro será tratada binariamente para cada tipo, ou seja, ou uma instância apresenta dado tipo de erro, ou não apresenta. As implementações perfeitas, sem ocorrência de nenhum tipo de erro, serão tidas como maior métrica de sucesso para um modelo. De modo similar, serão identificadas implementações que apresentem erros que impossibilitem a execução do jogo como medida de fracasso.

4. Resultados Esperados, Conclusão e Trabalhos Futuros

Neste trabalho, detalhamos a metodologia que usaremos para avaliar a capacidade de três LLMs de implementarem código em Python para jogos de tabuleiro, baseado no seu conhecimento prévio sobre eles. Também testaremos a capacidade dos modelos de raciocinar em alto nível sobre as regras do jogo implementado, pedindo variações dos jogos descritas em linguagem natural. Adicionalmente, verificaremos os efeitos do uso da Boardwalk API nos códigos gerados.

Como extensão de [Becker et al. 2025], os resultados deste estudo se somariam aos daquele na verificação de quais abordagens de prompting garantem maior taxa de sucesso na geração de código. Resultados positivos sobretudo nos testes com variações de regras representariam maior capacidade para a geração de código para jogos obscuros ou inéditos. Esperamos que o Claude novamente demonstre o melhor desempenho dentre

os três modelos. Entretanto, é possível que o sucesso médio de todos os modelos seja maior, visto que os prompts usados neste trabalho são mais simples e admitem maior de uso de conhecimento prévio dos jogos pelos modelos em relação à informação nova contida no prompt.

Trabalhos futuros podem abordar aspectos que ficariam em aberto na proposta, como a execução automatizada de playtests e a implementação de jogos completamente novos, fora do conhecimento prévio dos LLMs.

Agradecimentos

Este estudo foi financiado em parte pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001 e pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).

Referências

- Anthropic (2024). The Claude 3 Model Family: Opus, Sonnet, Haiku. https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf. [Accessed 21-04-2025].
- Becker, Á. G., de Oliveira, G. B., Rossato, L. B., and Tavares, A. R. (2025). Boardwalk: Towards a Framework for Creating Board Games with LLMs. In *SBGames*. To appear. Pre-print: <https://arxiv.org/abs/2508.16447v1>.
- Browne, C. (2011). *Evolutionary Game Design*. SpringerBriefs in Computer Science. Springer London, London, UK.
- Coignon, T., Quinton, C., and Rouvoy, R. (2024). A Performance Study of LLM-Generated Code on Leetcode. In *Proceedings of the 28th EASE*, pages 79–89.
- Genesereth, M., Love, N., and Pell, B. (2005). General game playing: Overview of the AAAI competition. *AI magazine*, 26(2):62–62.
- Liu, A. et al. (2025). DeepSeek-V3 Technical Report.
- Nielsen, J. L., Jensen, B. F., Mahlmann, T., Togelius, J., and Yannakakis, G. N. (2014). *AI for General Strategy Game Playing*, chapter 10, pages 274–304. John Wiley & Sons, Ltd.
- OpenAI et al. (2024). GPT-4 Technical Report.
- Piette, E., Soemers, D., Stephenson, M., Sironi, C., Winands, M., and Browne, C. (2020). Ludii – the ludemic general game system. *Frontiers in Artificial Intelligence and Applications*, 325:411–418.
- Rossato, L. B., Bombardelli, L. B., and Tavares, A. R. (2023). Evolutionary Tabletop Game Design: A Case Study in the Risk Game. In *SBGames*, page 161–170, New York, NY, USA. Association for Computing Machinery.
- Tanaka, T. and Simo-Serra, E. (2024). Grammar-based game description generation using large language models. *IEEE Transactions on Games*, pages 1–14.
- Todd, G., Padula, A. G., Stephenson, M., Piette, É., Soemers, D., and Togelius, J. (2024). GAVEL: Generating games via evolution and language models. *NeurIPS*, 37:110723–110745.