GERAÇÃO PROCEDURAL DE NÍVEIS 2D UTILIZANDO GRANDES MODELOS DE LINGUAGEM

PROCEDURAL GENERATION OF 2D LEVELS USING LARGE LANGUAGE MODELS

Marcelo Carlos da Fonseca Júnior[1]
Luiz Fernando Nunes[2]

RESUMO

Este estudo explora a aplicação de Grandes Modelos de Linguagem (LLMs) para Geração Procedural de Conteúdo (PCG) em jogos digitais, especificamente na criação de níveis para um jogo 2D semelhante ao clássico The Legend of Zelda. A metodologia inclui o fine-tuning do modelo LLaMA 3.1 com o dataset Zelda-Level-Games, adaptando-o para a geração de níveis jogáveis e personalizados através da linguagem natural, sendo possível a manipulação do design do nível a partir do input de entrada. A avaliação foi conduzida com métricas de jogabilidade, novidade, diversidade e precisão, demonstrando que o modelo atinge bons resultados na maioria das métricas, embora apresente desafios na precisão, que deixam ideias para trabalhos futuros. Os resultados indicam que o uso de LLMs para PCG é promissor, especialmente por possibilitar a criação de conteúdo customizado via linguagem natural, contribuindo para a acessibilidade e produtividade no design de jogos.

Palavras-chave: PCG; LLM; jogos; IA; personalizados; prompt.

ABSTRACT

This study explores the use of Large Language Models (LLMs) for Procedural Content Generation (PCG) in digital games, specifically for creating levels for a 2D game similar to the classic The Legend of Zelda. The methodology includes the fine-tuning of the LLaMa 3.1 model using the Zelda-Level-Games dataset,adapting it for the generation of playable and customized levels through natural language, allowing level design manipulation based on input instructions. The evaluation was conducted using metrics of playability, novelty, diversity, and accuracy, showing that the model achieves good results in most metrics, although it faces challenges in accuracy, suggesting areas for future work. The results indicate that using LLMs for PCG is promising, especially as it enables customized content creation through natural language, contributing to accessibility and productivity in game design

Keywords: PCG; LLM; games; AI; custom; prompt

[1] O nome do autor deve ser inserido de forma direta Prenome e Sobrenome. Deve constar o currículo sucinto de cada autor, com vinculação corporativa e endereço de contato em Notas de rodapé
[2] Idem ao anterior.

1 INTRODUÇÃO

O desenvolvimento dos níveis de um jogo é um processo demorado e caro, que envolve um grande esforço da equipe, pois o layout dos níveis é uma parte central da experiência do jogador. Com a evolução da tecnologia, a demanda por jogos com cada vez mais conteúdo aumentou drasticamente, o que pode sobrecarregar designers e desenvolvedores, levando ao estouro de prazos e orçamentos. A Geração Procedural de Conteúdo (PCG) surge como uma forma de lidar com esses desafios, utilizando algoritmos capazes de criar conteúdo automaticamente, reduzindo custos e tempo de desenvolvimento, mantendo a qualidade e adicionando diversidade e rejogabilidade (Hendrikx *et al*., 2013).

Diferentes algoritmos têm sido amplamente explorados para implementar a Geração Procedural de Conteúdo (PCG). Entre as abordagens mais tradicionais, destacam-se o Perlin Noise, comumente utilizado para a geração de terrenos (Kelly; Mccabe, 2006). O Cellular Automata também são amplamente empregados na criação de labirintos e, mais recentemente, adaptados para a geração de níveis (Adams; Parekh; Louis, 2017). Já as Cadeias de Markov possuem aplicações versáteis, mas se destacam na geração de níveis em jogos 2D, como para o jogo Super Mario Bros (Snodgrass; Ontañón, 2014). Nos últimos anos, o uso de técnicas de Inteligência Artificial (IA) na geração procedural ganhou destaque, especialmente com o surgimento de estudos envolvendo Aprendizado de Máquina (Summerville *et al*., 2018) e Aprendizado por Reforço (Khalifa *et al*., 2020), que têm apresentado bons resultados e aberto novas possibilidades para o campo da PCG.

Apesar do avanço das técnicas de Geração Procedural de Conteúdo (PCG), uma das principais limitações dos métodos atuais é a dificuldade em personalizar o conteúdo gerado. Adaptar o algoritmo para criar níveis com características específicas, como padrões de dificuldade, estilos visuais ou layouts únicos, é uma tarefacomplexa que frequentemente exige mudanças intrincadas na lógica do algoritmo e nos parâmetros de geração. Além disso, as pesquisas sobre este tema são limitadas, porém esse é um tema importante que pode afetar diretamente a experiência do jogador (Sudhakaran *et al*., 2024).

Para criar um método mais flexível e personalizável para a Geração

Procedural de Conteúdo (PCG), a utilização de Grandes Modelos de Linguagem (LLMs) se apresenta como uma solução promissora. De acordo com o seguinte estudo Zhao *et al*. (2023), LLMs referem-se a modelos de linguagem baseados na arquitetura Transformer que possuem centenas de bilhões (ou mais) de parâmetros e são treinados em grandes quantidades de dados textuais. Esses modelos são especialmente poderosos para compreender a linguagem natural e resolver tarefas complexas por meio da geração de texto. Por conta desse alto potencial, os LLMs têm ganhado destaque no mercado da tecnologia. O lançamento do ChatGPT pela OpenAI (OpenAI, 2024) em 2022, por exemplo, consolidou a importância dessas ferramentas, atraindo a atenção de grandes empresas que desenvolveram seus próprios LLMs, como a Google que desenvolveu o Gemini (Team *et al*., 2023), e Meta que lançou o Llama (Touvron *et al*., 2023).

Embora os LLMs já apresentem uma capacidade supreendente em sua forma "base", eles ainda podem ser aprimorados por meio do processo de fine-tuning. No caso de diversos modelos como o Llama e o GPT, esse processo envolve o uso de três parâmetros: instruction, input e output. O instruction fornece diretrizes gerais sobre o que o modelo deve gerar, enquanto o input adiciona um contexto mais específico, detalhando as características particulares da tarefa. O output representa o resultado desejado que o modelo deve aprender a gerar. Ao ajustar suas previsões com base nesses pares, o modelo refina seus parâmetros para que, ao receber novos inputs similares no futuro, possa gerar respostas mais adequadas à tarefa específica (Zhao *et al*., 2023).

No contexto da PCG, os LLMs podem ser treinados com dados de um jogo 2D, onde os níveis seriam representados em forma de texto, como uma matriz bidimensional. Cada célula da matriz corresponde a um elemento do jogo, como blocos ou inimigos, e essa matriz é transformada em uma sequência linear, utilizando tokens especiais para separar as linhas da matriz. Junto ao instruction e input que descrevem o layout, ofuncionamento geral do jogo e as propriedades específicas de cada nível, assim o modelo pode "aprender"padrões e características, além de entender aspectos como o objetivo e a jogabilidade do jogo. Dessa forma, o fine-tuning não apenas permite que o modelo gere novos níveis automaticamente, mas também permite uma personalização precisa, como ajustar a dificuldade, controlar a

quantidade de certos blocos ou elementos, e até mesmo criar layouts específicos com base nas instruções fornecidas (Todd *et al*., 2023).

Uma das grandes vantagens dessa abordagem é que, após o treinamento do modelo, a geração de níveis pode ser feita inteiramente através de comandos em linguagem natural. Isso elimina a necessidade de interação direta com código, tornando a PCG mais acessível para profissionais que não têm experiência com programação, como designers de jogos.

Ainda assim, é fundamental reconhecer alguns desafios técnicos. Entre eles, destaca-se a dificuldade dos LLMs em representar adequadamente a estrutura espacial dos níveis de jogos, que possuem uma natureza bidimensional ou tridimensional, enquanto os LLMs processam dados de forma linear. Além disso, os LLMs dependem de grandes volumes de dados para treinamento, e datasets específicos de níveis jogáveis nem sempre estão amplamente disponíveis (Todd *et al*., 2023).

Para explorar o potencial dos LLMs neste contexto, este artigo propõe o \textit{fine-tuning} de um modelo para gerar níveis em um jogo 2D similar a The Legend of Zelda, utilizando o dataset Zelda-Game-Leve (Zafar, 2023),, que contém mais de 1000 níveis jogáveis. O modelo será baseado no Llama 3.1 8b. Esse modelo passará pelo processo de fine-tuning e então será avaliado quanto à sua capacidade de gerar níveis jogáveis e personalizados, com o intuito de checar a viabilidade de utilizar LLMs como uma ferramenta prática para PCG.

2 FUNDAMENTAÇÃO TEÓRICA

A Geração Procedural de Conteúdo (PCG) desempenha um papel essencial na criação de jogos digitais, automatizando a produção de elementos como cenários, personagens e níveis, permitindo que jogos sejam mais diversificados e personalizados. Esse processo, conforme detalhado por Togelius *et al*. (2013), envolve não apenas a geração de conteúdos de um jogo, mas também a criação de experiências complexas que se alinham ao estilo de cada jogador, apresentando experiências únicas. Para Hendrikx *et al*. (2013), a PCG surge como uma resposta ao desafio de escalabilidade que a indústria enfrenta ao tentar produzir conteúdo de alta

qualidade para um público em constante expansão, ajudando a reduzir os custos de desenvolvimento em prazos curtos. Complementando essa visão, Smith (2015) explora as raízes do PCG nos jogos analógicos, destacando como sistemas de regras e modularidade já permitiam a criação de conteúdo muito antes da era digital, ressaltando como esta é uma ferramente presente a muito tempo na indústria de jogos.

As técnicas de Geração Procedural de Conteúdo (PCG) englobam métodos variados e adaptados conforme o contexto e o tipo de conteúdo desejado. Algumas técnicas clássicas utilizadas para o desenvolvimento de níveis para jogos são as abordagens baseadas em algoritmos genéticos, como são utilizadas em Togelius *et al*. (2011) que discute uma abordagem baseada em busca para geração procedural de níveis no estilo de labirintos. Diversos outros estudos também utilizam algoritmos evolutivos combinados com outros métodos, com a finalidade de gerar níveis, como o Baghdadi *et al*. (2015) e Gellel e Sweetser (2020) que utilizam esses algoritmos de formas diferentes para geração de níveis para o jogo spelunky e jogos do gênero roguelike, respectivamente.

Abordagens utilizando técnicas de Inteligência Artificial se tornaram muito populares recentemente, no estudo PCGML Summerville *et al*. (2018), explora aPCG com uso de aprendizado de máquina, focando em como modelos treinados com conteúdo existente podem criar novos níveis, mapas, histórias interativas e até itens colecionáveis para jogos, são discutidas técnicas como redes neurais (LSTM, autoencoders), cadeias de Markov, e fatores de matriz, além de desafios como aprendizado com conjuntos de dados pequenos e geração de conteúdo personalizável. Técnicas de Aprendizado por Reforço também obtiveram destaque, principalmente com o artigo PCGRL (Khalifa *et al*., 2020) que propõe uma abordagem iterativa para o design de níveis, onde agentes de aprendizado por reforço tomam decisões sequenciais para melhorar a qualidade final do nível, a partir desta ideia também foi desenvolvido o ARLPCG (Gisslén *et al*., 2021), apresentando um modelo que usa aprendizado por reforço adversarial para gerar ambientes de jogos 3D, aliando dois agentes, um gerador, que cria os níveis e um solucionador, que joga os níveis, com os dois agentes dentro de um loop de feedback, assim sendo capazes de gerar níveis adaptativos e dinâmicos.

Os Grandes Modelos de linguagem (LLMs) revolucionaram o campo da inteligência artificial, com capacidade para realizar tarefas complexas e entendimento profundo de linguagem, os LLMs alcançaram uma capacidade sem precedentes através do aumento de parâmetros e dados de treinamento (Zhao *et al*., 2023). Trazendo inovações para diversos setores de pesquisa, como na medicina, onde foi discutido o uso de modelos como o ChatGPT, para suporte em tarefas clínicas, educativas e de pesquisa, o estudo sugere que, embora LLMs possam melhorar a eficiência no setor, são necessárias mais regulamentações para seu uso efetivo (Thirunavukarasu *et al*., 2023), LLMs também foram testadas para aumento da produtividade de programadores, onde o estudo realizou testes com 32 participantes, que mostraram resultados positivos na taxa de conclusão de tarefas utilizando LLMs (Nam *et al*., 2024).

Dentro do setor de desenvolvimento de jogos, o uso de LLMs é muito promissor para a PCG, modelos como o MarioGPT, que foi desenvolvido a partir do GPT-2 e personalizado com cross-attention, foi capaz de gerar ótimos níveis para o jogo Super Mario Bros de forma diversificada e personalizável (Sudhakaran *et al*., 2024), outro jogo popular para testes com PCG é o sokoban, um jogo de puzzles complexos, que foi utilizado no estudo de Togelius (Todd *et al*., 2023), onde um modelo do GPT-2 e GPT-3, a partir do processo de fine-tuning, foram capazes de gerar níveis para o jogo, com estes testes foram estudados os efeitos de pré-treinamento dos modelos e do tamnho dos datasets utilizados, além de indicar um grande potencial dos novos modelos, como o GPT-3 e seus sucessores. Por último, o artigo (Taveekitworachai *et al*., 2023), que descreve uma competição que incentiva o uso do ChatGPT para gerar níveis do jogo Science Birds (similar ao Angry Birds) a partir de prompts, mostrando como apenas a engenharia de prompts é capaz de gerar níveis complexos sem uso do fine-tuning.

3 METODOLOGIA

Este capítulo apresenta a metodologia utilizada para desenvolver um modelo de Geração Procedural de Conteúdo (PCG) utilizando um Grande Modelo de Linguagem (LLM) para criar níveis de um jogo 2D similar ao The Legend of Zelda

original. São detalhados o processo de coleta e preparação de dados, o treinamento do modelo, a geração dos níveis em formato de texto, e a avaliação dos resultados gerados.

O jogo escolhido para este trabalho é uma versão simplificada do clássico The Legend of Zelda lançado originalmente para o Nintendo Entertainment System (Nintendo, 1986). O jogo é representado em uma grid de 13x9, cercada por paredes, onde o jogador (avatar) deve navegar para completar o objetivo. Em cada nível, o avatar precisa coletar uma chave e levá-la até uma porta para vencer, enquanto evita ou combate inimigos espalhados pelo mapa. Esses inimigos variam em velocidade e comportamento, o que acrescenta desafios e exige diferentes estratégias por parte do jogador.

3.1 DESCRIÇÃO DO JOGO

O jogo escolhido para este trabalho é uma versão simplificada do clássico The Legend of Zelda lançado originalmente para o Nintendo Entertainment System (Nintendo, 1986). O jogo é representado em uma grid de 13x9, cercada por paredes, onde o jogador (avatar) deve navegar para completar o objetivo. Em cada nível, o avatar precisa coletar uma chave e levá-la até uma porta para vencer, enquanto evita ou combate inimigos espalhados pelo mapa. Esses inimigos variam em velocidade e comportamento, o que acrescenta desafios e exige diferentes estratégias por parte do jogador.

Cada nível do jogo é representado de forma similar à representação dos níveis de The Legend of Zelda feita pelo Video Game Level Corpus (VGLC) (Summerville *et al*., 2016), que padroniza a representação de conteúdo em estudos sobre geração procedural de jogos. A escolha dessa padronização visa facilitar o processamento dos níveis e garantir consistência com pesquisas anteriores, permitindo o uso eficiente dos dados no treinamento do modelo de geração procedural.

Essa versão do jogo foi disponibilizada pela GVGAI (General Video Game Artificial Intelligence), uma framework que oferece suporte a mais de 100 jogos 2D e inclui diversos agentes inteligentes capazes de jogar automaticamente esses jogos. A GVGAI é uma ferramenta valiosa para testes de níveis 2D, pois consegue interpretar

arquivos de texto que seguem um formato similar ao Video Game Level Corpus (VGLC) e utilizá-los como níveis jogáveis dentro dos jogos especificados. A framework também permite que os níveis sejam testados tanto por agentes inteligentes quanto por jogadores humanos, oferecendo uma maneira eficaz de validar a jogabilidade dos níveis gerados (Perez-Liebana *et al*., 2019).

3.2 COLETA E PREPARAÇÃO DE DADOS

Os dados para o treinamento do modelo foram obtidos a partir do dataset Zelda-Game-Levels disponível no Kaggle (Zafar, 2023). Este dataset contém 2048 arquivos de imagem de níveis do jogo Zelda da GVGAI, dos quais 1024 são considerados jogáveis e 1024 são considerados injogáveis.

O dataset original consiste em arquivos de imagem, mas, para os objetivos deste estudo, os níveis precisaram ser convertidos para o formato textual VGLC, pois os LLMs são projetados para trabalhar com entradas textuais.

Figura 1 – Nível retirado do dataset

Fonte: Zafar (2023)

3.2.1 Conversão das Imagens para Texto

Para realizar essa conversão, foi necessário processar as imagens e identificar cada tile presente no nível com base em sua cor. O procedimento foi realizado em várias etapas:

Análise de cor dos tiles: Cada variação de tile foi analisada com base na sua cor. Para isso, a média de cor de cada elemento (como paredes, jogador, inimigos, chave e porta) foi calculada e associada ao tile correspondente.

Divisão do nível em grid: Cada nível foi dividido em uma grade (grid), onde cada quadrado representa um tile no jogo. Os quadrados foram então comparados com os valores de cor estabelecidos para determinar o elemento que eles representavam (por exemplo, paredes, espaços vazios, inimigos, etc.).

Conversão para caracteres: Com base nessa análise de cor, cada tile foi categorizado e transformado em um caractere correspondente, seguindo a convenção VGLC.

Então, originalmente os níveis eram arquivos de imagem como mostra a Figura 1, depois de fazer a associação das cores com os carácteres correspondentes, o nível então foi transformado numa matriz, como a da tabela abaixo, que é uma representação direta do nível da figura

Tabela 1 – Representação textual do nível

W	W	W	W	W	W	W	W	W	W	W	W	W
W	-	-	-	-	-	-	-	-	-	-	2	W
W	-	-	-	-	-	-	-	-	-	-	-	W
W	-	-	-	D	-	+	-	-	3	-	-	W
W	-	-	-	-	-	-	-	-	-	B	1	W
W	-	-	-	-	-	-	-	-	-	D	-	W
W	-	-	-	-	-	-	-	-	-	-	-	W
W	2	A	-	-	-	-	-	1	-	B	1	W
W	W	W	W	W	W	W	W	W	W	W	W	W

Fonte: Autoria Própria

Tabela 2 – Legenda dos símbolos utilizados no nível

Cores e Caracteres	Significado
W	Parede
A	Jogador (Avatar)
-	Espaço vazio
1	Inimigo lento
2	Inimigo comum
3	Inimigo rápido
+	Chave
D	Porta

Fonte: Autoria Própria

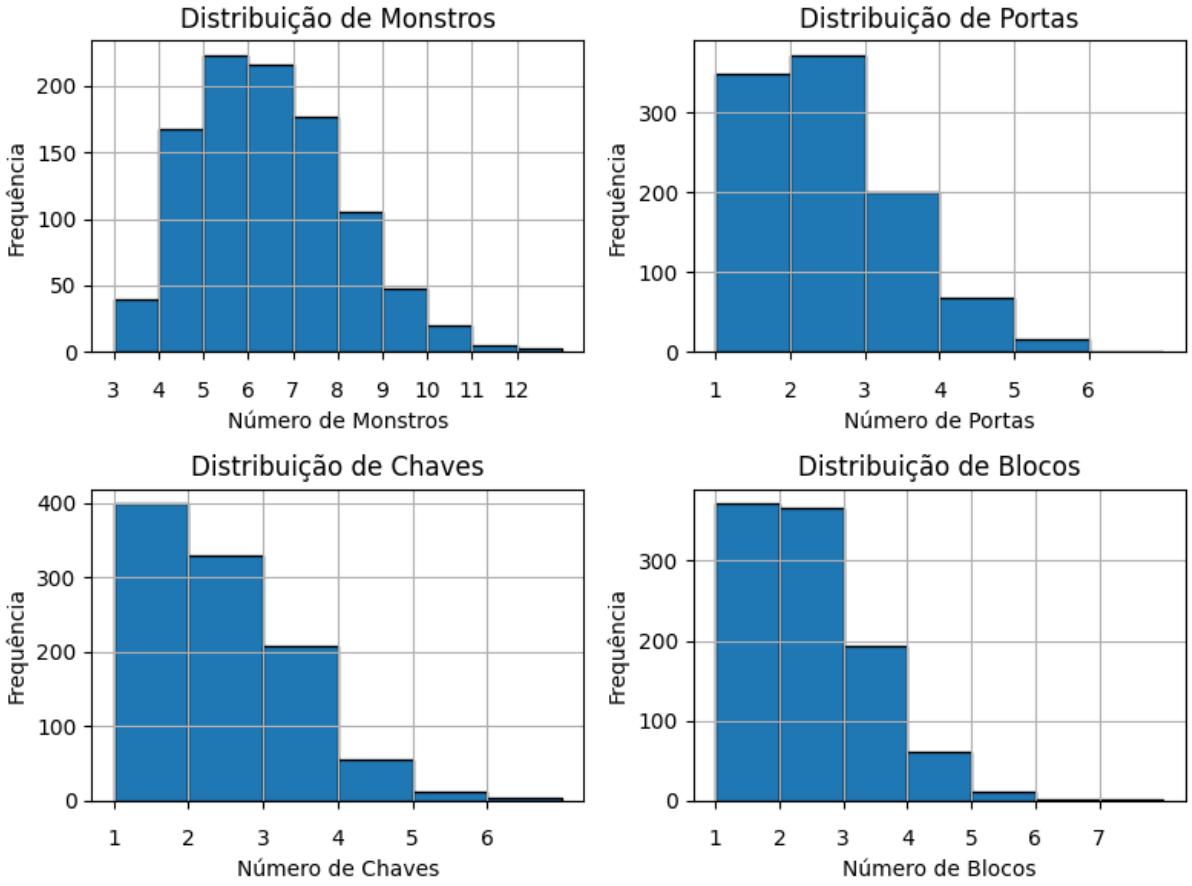
3.2.2 Criação dos inputs

Após a conversão das imagens para os textos tokenizados, foi necessário criar instruções que seriam passadas para o fine-tuning dos modelos, essas instruções precisariam descrever as propriedades únicas de cada nível. Como o jogo escolhido é relativamente simples, o que varia de um nível para o outro é a quantidade de inimigos, quantidade de blocos presentes dentro do nível, quantidade de portas e de chaves. Então cada input deveria especificar relativamente a quantidade de cada um desses elementos.

Para o desenvolvimento dos inputs, foi feita uma análise dos níveis do dataset. Os gráficos presentes na Figura 1 mostram a distribuição da quantidade de cada elemento e a frequência em que certa quantidade aparece durante todo os níveis.

Com essas informações, uma função foi capaz de comparar as quantidades de cada elemento em cada nível com a quantidade média de cada elemento, criando um input que listava a quantidade de blocos, inimigos, portas e chaves o nível deveria ter. Gerando assim inputs que indicam as características individuais de cada nível, facilitando a personalização através da linguagem natural.

Figura 2 – Gráficos da frequência de cada elemento



Fonte: Autoria Própria

3.3 FINE-TUNING DOS MODELOS

O processo de fine-tuning do modelo LLAMA 3.1 foi otimizado com o uso do Unsloth (Unsloth, 2024), uma ferramenta de código aberto que permite acelerar o treinamento em até 5 vezes e reduzir o consumo de memória em 80%. O Unsloth consegue fazer essa otimização através do uso de Kernels personalizados que aceleram os cálculos feitos durante o fine-tuning e com o uso de técnicas como quantization e Low-Rank Adaptation (LoRA) para reduzir o uso de memória. Além de ter uma configuração personalizada para o GPU disponível no Google Colab, facilitando o treinamento nessa máquina.

Essa otimização foi extremamente importante, pois o equipamento disponível tinha poder computacional limitado. O uso da GPU do Google Colab, combinado com

as otimizações do Unsloth, possibilitou que o fine-tuning fosse realizado de maneira rápida e eficiente, superando as limitações de hardware.

Para o fine-tuning do modelo baseado no Meta-Llama-3.1-8B-bnb-4bit, denominado de llama-level-generator, foram passados os três parâmetros para o finetuning, um instruction que descrevia o layout geral do jogo, indicando o formato do nível de 13x9, o significado de cada token, e os requisitos mínimos para um nível ser jogável, o input trazendo um contexto maior de cada nível, indicando a quantidade de cada elemento, e o output com o nível.

O processo de treino foi o recomendado pelo unsloth, com uma época, 126 passos e utilizando o tokenizador padrão para o Llama.

3.4 AVALIAÇÃO

A avaliação do modelo foi realizada utilizando um conjunto de métricas utilizadas por Todd *et al*. (2023), essas métricas abrangem diversas áreas da geração do modelo, facilitando a avaliação dos aspectos mais importantes e indicando espaços para possíveis melhorias. Para isso, foram definidos quatro critérios principais: jogabilidade, novidade, diversidade e precisão.

3.4.1 Jogabilidade

Para avaliar a jogabilidade, foi utilizada uma abordagem baseada no agente ASTAR disponível na framework GVGAI, este agente é capaz de jogar de maneira automática e finalizar a grande maioria dos níveis por conta própria. O objetivo dessa métrica é verificar se os níveis gerados são jogáveis de acordo com as regras do jogo implementado neste trabalho. O nível é considerado jogável se: Contém uma chave e uma porta; For solucionável pelo agente ASTAR, sendo capaz de encontrar uma solução viável em até 2000 passos. Caso o agente não encontre uma solução nesse limite, o nível é classificado como injogável.

Essa métrica avalia a funcionalidade mínima dos níveis e é crucial para garantir que os conteúdos gerados possam ser utilizados sem configuração externa.

3.4.2 Novidade

A métrica de Novidade mede o quão únicos são os níveis gerados em comparação com os presentes no conjunto de treinamento. Para isso, a avaliação de novidade foi feita com base na distância de levenshtein (ou distância de edição), esta é uma métrica que quantifica o número mínimo de operações (inserções, deleções e substituições) necessárias para transformar uma string em outra. Esta métrica será aplicada entre as strings que representam os níveis do treinamento e os níveis gerados pelo modelo, utilizando o mesmo input. Dois níveis são considerados distintos se a distância de edição entre eles for maior que um mínimo predefinido, no caso desse estudo será utilizado k = 5. Este limite foi ajustado para garantir que pequenas modificações em um nível não sejam contabilizadas como novidade.

Essa métrica foi utilizada para assegurar que os níveis gerados não fossem apenas cópias dos dados de treinamento, e garantir que o modelo estivesse gerando seus próprios níveis.

3.4.3 Diversidade

A diversidade, neste estudo, é a diferença entre os níveis gerados pelo mesmo input, essa métrica é importante para garantir que os níveis gerados sejam diferentes entre si, e assegurar que os modelos não estejam generalizando os seus dados.

A avaliação da diversidade dos níveis gerados foi realizada com base na distância de edição, similar a avaliação de novidade, porém adicionando uma abordagem de grafos. Primeiro, é considerado que dois níveis são distintos se a distância de edição entre eles for maior que um valor limite (k = 5). Então, os níveis gerados são representados como um grafo, onde cada nó corresponde a um nível, e existe uma aresta entre dois nós apenas se a distância de edição entre eles for maior que 5. A Taxa de Diversidade é determinada pela razão entre o tamanho do maior clique identificado (conjunto de nós onde todos estão conectados entre si) e o número total de níveis gerados

$$Diversidade = \frac{Tamanho\ do\ Maior\ Clique}{Número\ Total\ de\ Níveis}$$

3.4.4 Precisão

A precisão foi medida para verificar a aderência dos níveis gerados às instruções fornecidas no input de entrada. Os inputs detalhavam características específicas do nível, como o número de inimigos, blocos, chaves e portas. A precisão foi calculada comparando-se as características dos níveis gerados com aquelas definidas no input.

Para cada nível gerado, foi verificado se ele seguia as instruções com uma tolerância pré-definida. A precisão é fundamental para avaliar a capacidade dos modelos de gerar níveis personalizáveis com base nas características fornecidas em linguagem natural.

4 RESULTADOS

4.1 NOVIDADE

Utilizando a métrica de distância de edição com valor limite k=5, foi possível avaliar a capacidade dos modelos de gerar níveis verdadeiramente novos. Todos os 1200 níveis gerados apresentaram uma distância de edição suficiente para serem considerados distintos quando comparados com os níveis do dataset original.

4.2 DIVERSIDADE

Os modelos também mostraram uma alta capacidade de diversidade, com 100% dos níveis gerados sendo considerados distintos tanto entre si quanto entre níveis de um mesmo input. Essa diversidade se manifesta em diferentes aspectos do layout dos níveis, como o posicionamento de inimigos, blocos e itens, e na quantidade desses elementos.

4.3 JOGABILIDADE

Já a jogabilidade foi avaliada utilizando o agente ASTAR dentro da framework GVGAI, que utilizou 3 tentativas de no máximo 2000 ações para testar cada nível e determinar se eles poderiam ser vencidos. Dos 1200 níveis gerados, 976 foram classificados como jogáveis, enquanto 224 foram considerados injogáveis. Os principais problemas encontrados nos níveis injogáveis foram a ausência de uma chave ou do avatar (jogador), o que impossibilita a finalização do nível.

Tabela 3 – Quantidade de níveis jogáveis e não jogáveis

Resultado	**Quantidade**
Níveis Jogáveis	81.3%
Níveis Não Jogáveis	18.6%

Fonte: Autoria Própria

4.4 PRECISÃO

A precisão dos modelos foi analisada comparando as características dos níveis gerados com as especificações fornecidas nos inputs. Dos 1200 níveis, 586 foram considerados precisos, enquanto 614 foram classificados como imprecisos. Esse resultado indica que, embora o modelo consiga atender parcialmente aos inputs, ainda sofre em seguir inputs mais específicos.

Tabela 4 – Distribuição da Acurácia

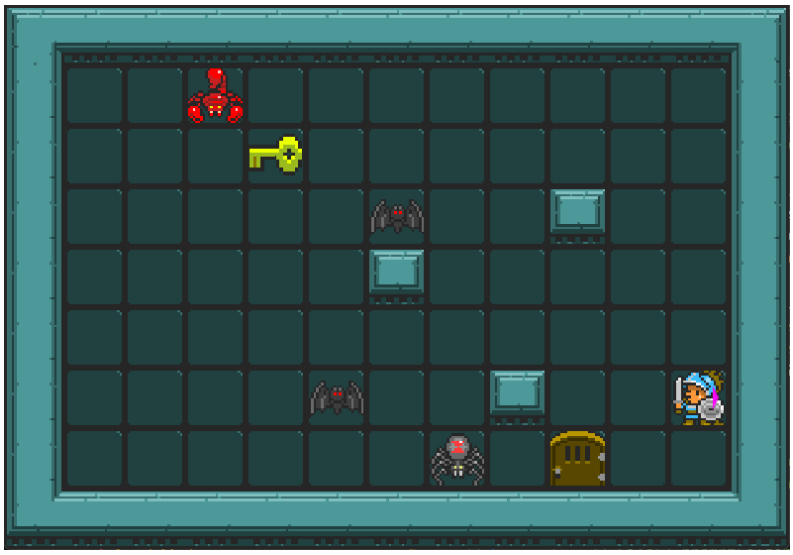
Resultado	**Quantidade**
Níveis Precisos	48.8%
Níveis Imprecisos	51.1%

Fonte: Autoria Própria

5 CONCLUSÃO

O Nível da Figura 3, mostra um exemplo de nível gerado pelo modelo que atendeu todos os critérios de qualidade, e realmente o nível gerado mostra muito potencial, apresentando desafio ao jogador pela quantidade de inimigos e a distância entre a chave e a porta, mas ainda assim se mantendo jogável e interessante.

Figura 3 – Nível gerado pelo modelo



Fonte: Autoria Própria

Este nível representa bem os resultados das métricas de Novidade e Diversidade, que tiveram um ótimo desempenho, estes resultados reforçam o poder do modelo LLaMA 3.1 em gerar conteúdo original, em contraste com modelos anteriores, como o GPT-2, que, em estudos prévios Todd *et al*. (2023), não alcançou o níveis similares de diversidade e novidade. Além disso, isto mostra uma alta capacidade de generalização dos dados utilizados para o fine-tuning, afirmando que o modelo esta realmente aprendendo os padrões e características dos níveis e não apenas replicando o que foi passado no fine-tuning. As métrica de jogabilidade também reforçam isso, pois assim como todos os níveis do dataset original foram testados e considerados jogáveis, o modelo conseguiu replicar um alta taxa de jogabilidade em seus níveis.

Por outra via, os resultados da métrica de precisão mostram algumas

limitações do modelo. Com uma taxa de acerto em torno de 48,8%, vemos que o modelo é parcialmente eficaz, conseguindo atender aproximadamente metade das solicitações. Esse índice de precisão sugere que há potencial para melhorias na forma como o modelo entende e executa as instruções dos inputs, especialmente em contextos onde o nível exige uma configuração mais específica.

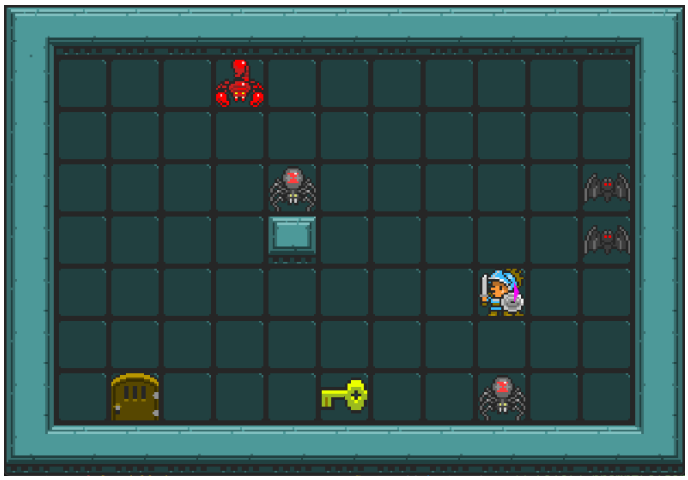
Tabela 5 – Frequência de erros encontrados nos níveis gerados

Erro	**Quantidade**
Chaves Insuficientes: 1 encontrada	40.4%
Inimigos presentes quando nenhum era esperado	33.2%
Portas Insuficientes: 1 encontrada	22.3%
Chave faltando	19%
Porta faltando	4.3%
Poucos inimigos: 4 encontrados	2.4%
Muitos inimigos: 7 encontrados	1.1%
Muitos inimigos: 8 encontrados	0.16%

Fonte: Autoria Própria

Os erros mais comuns, observados na 4, envolvem a falta de elementos em alguns níveis, como chaves e portas, ou a presença de inimigos quando o input não solicita.Por exemplo, o problema mais frequente, "Chaves Insuficientes: 1 encontrada"e o terceiro mais frequente, "Portas Insuficientes: 1 encontrada", indica que o modelo acaba inserindo apenas uma porta ou chave por nível, mesmo que o input especifique mais, isso pode acontecer pois a instruction que é passada para todo os níveis, indica que um nível jogável precisa de pelo menos uma porta e uma chave, essa especificação pode confundir o modelo que acaba por deixar apenas uma porta ou chave.

Figura 4 – Nível com apenas uma porta e uma chave



Fonte: Autoria Própria

Figura 5 – Nível com inimigos

Fonte: Autoria Própria

Outro erro comum foi a presença de inimigos quando nenhum era esperado, com um certo nível de incerteza com o dataset utilizado para o fine-tuning do modelo, pois dentre os 1024 níveis apenas 44 possuem nenhum inimigo, dificultando a generalização dessa especificação e acarretando em alguns problemas de precisão nesse quesito

Os níveis na Figura 4 e Figura 5, mostram exemplos de níveis considerados imprecisos pelos motivos citados anteriormente, o primeiro exemplo é um nível que o input dizia "Crie um nível com poucos blocos, muitos inimigos, muitas portas e muitas chaves", o resultado foi um nível bom, considerado novo, diverso e jogável, que

atende os critérios de poucos blocos e muitos inimigos, porém possui apenas uma chave a uma porta, mesmo sendo especificado no input o uso de muitas chaves e muitas portas.

O mesmo acontece no segundo exemplo, onde o input "Crie um nível com muitos blocos, nenhum inimigo, uma porta e uma chave"é utilizado, e de forma similar ao exemplo anterior, todos os critérios de qualidade foram atendidos, e a maioria das especificações de personalização também, menos o comando de "nenhum inimigo".

Apesar de um resultado com valores abaixo quando comparado às outras métricas, a precisão do modelo ainda pode ser considerada muito boa, pois houve uma generalização da maioria das propriedades do input, e o modelo foi capaz de entender bem a diferença entre poucos e muitos, já que quase não houveram problemas com essas características, somente poucos casos em que o modelo acrescentava ou retirava um inimigo a mais, além de não ocorrer nenhum problema sequer com a falta ou exagero de blocos.

Todos esses resultados são valiosos para o refinamento do modelo, sugerindo possíveis melhorias na engenharia de prompt, como enfatizar a quantidade dos elementos, usar inputs mais detalhados ou adicionar instructions personalizadas para cada nível. Além da utilização de um dataset maior e mais diverso, podendo gerar mais exemplos para o aprendizado do modelo, porém este é um problema comum quando se trata de PCG e desenvolvimento de jogos.

Mesmo com as limitações observadas, este trabalho contribui de forma significativa para a comunidade de desenvolvimento de jogos, mostrando o grande potencial da nova geração de grandes modelos de linguagem, como o LLaMA 3.1, na Geração Procedural de Conteúdo (PCG). Através dos testes realizados, comprovou-se que esses modelos podem adaptar-se às especificações fornecidas em linguagem natural, o que representa um avanço considerável em termos de personalização e controle humano na criação de níveis, promovendo uma maior diversidade criativa na indústria.

Essa pesquisa também facilita o desenvolvimento de jogos para profissionais sem experiência em programação que agora conseguem utilizar ferramentas que antes eram inacessíveis. Além de aumentar o desempenho das equipes de desenvolvedores, que na indústria atual sofrem constantemente com a quantidade de

conteúdos para serem desenvolvidos em curtos prazos.

REFERÊNCIAS

ADAMS, C.; PAREKH, H.; LOUIS, S. J. Procedural level design using an interactive cellular automata genetic algorithm. *In*: PROCEEDINGS OF THE GENETIC AND EVOLUTIONARY COMPUTATION CONFERENCE COMPANION, 17., 2017. **Proceedings** [...]. [S.l.: s.n.], 2017. p. 85–86.

BAGHDADI, W. *et al*. A procedural method for automatic generation of spelunky levels. *In*: EUROPEAN CONFERENCE ON THE APPLICATIONS OF EVOLUTIONARY COMPUTATION, 18., 2015, Copenhagen, Denmark. **Proceedings** [...]. Copenhagen, Denmark: Springer, 2015. p. 305–317.

GELLEL, A.; SWEETSER, P. A hybrid approach to procedural generation of roguelike video game levels. *In*: INTERNATIONAL CONFERENCE ON THE FOUNDATIONS OF DIGITAL GAMES, 15., 2020, Bugibba Malta. **Proceedings** [...]. Bugibba Malta: ACM, 2020. p. 1–10.

GISSLÉN, L. *et al*. Adversarial reinforcement learning for procedural contente generation. In: IEEE. 2021 IEEE Conference on Games (CoG). [S.l.], 2021. p. 1–8.

HENDRIKX, M. *et al*. Procedural content generation for games: A survey. **ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM),** New York, NY, USA, v. 9, n. 1, p. 1–22, 2013.

KELLY, G.; MCCABE, H. A survey of procedural techniques for city generation. **The ITB Journal**, v. 7, n. 2, p. 5, 2006.

KHALIFA, A. *et al*. Pcgrl: Procedural content generation via reinforcement learning. *In*: AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE AND INTERACTIVE DIGITAL ENTERTAINMENT, 16., 2020. **Proceedings** [...]. [S.l.: s.n.], 2020. v. 16, n. 1, p. 95–101.

NAM, D. *et al*. Using an llm to help with code understanding. *In*: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 46., 2024, Lisboa. **Proceedings** [...]. Lisboa: ACM / IEEE, 2024. p. 1–13.

Nintendo. **The Legend of Zelda.** 1986. [Nintendo Entertainment System].

OpenAI. **Página Oficial da OpenAI**. 2024. Disponível em: https://openai.com. Acesso em: 27 out. 2024.

PEREZ-LIEBANA, D. *et al*. General video game AI: A multitrack framework for evaluating agents, games, and content generation algorithms. **IEEE Transactions on Games**, v. 11, n. 3, p. 195–214, 2019.

SMITH, G. **An analog history of procedural content generation**. Boston, MA: FDG, 2015. p. 1–6.

SNODGRASS, S.; ONTAÑÓN, S. Experiments in map generation using markov chains. *In*: INTERNATIONAL CONFERENCE ON FOUNDATIONS OF DIGITAL GAMES, 2014, Santiago. **Procedings** [...]. Santiago: FDG, 2014.

SUDHAKARAN, S. *et al*. Mariogpt: Open-ended text2level generation through large language models. **Advances in Neural Information Processing Systems**, v. 36, 2024.

SUMMERVILLE, A. *et al*. Procedural content generation via machine learning (pcgml). **IEEE Transactions on Games**, v. 10, n. 3, p. 257–270, 2018.

SUMMERVILLE, A. J. *et al*. The vglc: The video game level corpus. **arXiv preprint** arXiv:1606.07487, 2016.

TAVEEKITWORACHAI, P. *et al*. ChatGPT4PCG competition: character-like level generation for science birds. **arXiv,** arXiv:2303.1, mar. 2023.

TEAM, G. *et al*. Gemini: a family of highly capable multimodal models. **arXiv preprint** arXiv:2312.11805, 2023.

THIRUNAVUKARASU, A. J. *et al*. Large language models in medicine. **Nature medicine**, **Nature Publishing Group US**, New York, v. 29, n. 8, p. 1930–1940, 2023.

TODD, G. *et al*. Level generation through large language models. *In*: INTERNATIONAL CONFERENCE ON THE FOUNDATIONS OF DIGITAL GAMES, 18., 2023, Lisboa. **Proceedings** [...]. Lisboa: ACM, 2023. p. 1–8.

TOGELIUS, J. *et al*. Procedural content generation: Goals, challenges and actionable steps. *In*: LUCAS, S.M. *et al.* (Ed.). **Artificial and Computational Intelligence in Games** - Dagstuhl Fol low-Ups, volume 6. Germany: Schloss Dagstuhl-Leibniz-Zentrum Fuer Informatik, 2013. p. 61-75.

TOGELIUS, J. *et al*. Search-based procedural content generation: A taxonomy and survey. **IEEE Transactions on Computational Intelligence and AI in Games**, v. 3, n. 3, p. 172–186, 2011.

TOUVRON, H. *et al*. Llama: Open and efficient foundation language models. **arXiv preprint** arXiv:2302.13971, 2023.

UNSLOTH. **Unsloth.** 2024. Disponível em: https://unsloth.ai/. Acesso em: 10 out. 2024.

ZAFAR, A. **Zelda Game Levels.** 2023. Disponível em:

https://www.kaggle.com/datasets/adizafar/zelda-game-levels. Acesso em: 10 out. 2024.

ZHAO,W. X. *et al*. A survey of large language models. **arXiv preprint** arXiv:2303.18223, 2023.