Level Generation Through Large Language Models

Graham Todd
gdrtodd@nyu.edu
New York University Tandon
Brooklyn, New York, USA

Sam Earle
se2161@nyu.edu
New York University Tandon
Brooklyn, New York, USA

Muhammad Umair Nasir
umairnasir1@students.wits.ac.za
University of the Witwatersrand
Johannesburg, South Africa

Michael Cerny Green
mike.green@nyu.edu
New York University Tandon
Brooklyn, New York, USA

Julian Togelius
julian@togelius.com
New York University Tandon
Brooklyn, New York, USA

ABSTRACT

Large Language Models (LLMs) are powerful tools, capable of leveraging their training on natural language to write stories, generate code, and answer questions. But can they generate functional video game levels? Game levels, with their complex functional constraints and spatial relationships in more than one dimension, are very different from the kinds of data an LLM typically sees during training. Datasets of game levels are also hard to come by, potentially taxing the abilities of these data-hungry models. We investigate the use of LLMs to generate levels for the game *Sokoban*, finding that LLMs are indeed capable of doing so, and that their performance scales dramatically with dataset size. We also perform preliminary experiments on controlling LLM level generators and discuss promising areas for future work.

KEYWORDS

procedural content generation, sokoban, language models, transformers

ACM Reference Format:
Graham Todd, Sam Earle, Muhammad Umair Nasir, Michael Cerny Green, and Julian Togelius. 2023. Level Generation Through Large Language Models. In *Foundations of Digital Games 2023 (FDG 2023), April 12–14, 2023, Lisbon, Portugal*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3582437.3587211

1 INTRODUCTION

In recent years, attention-based large language models (LLMs) have taken the world by storm, demonstrating surprisingly high performance on a variety of natural language tasks. With the right tuning, LLMs have been shown to generate coherent text in a number of styles, produce working snippets of computer code, and even respond naturalistically to human questions and conversation. While the *architectures* underlying these models have been leveraged for tasks outside the realm of standard text generation, from music [6] to reinforcement learning [3], comparatively less effort has been spent on analyzing the capacity of the LLMs themselves to produce non-lingusitic artifacts while still leveraging their vast amounts of training data. In this paper, we investigate the ability of LLMs to generate video game levels and the extent to which truths about these models taken from natural language processing apply to this new domain. We also conduct preliminary experiments on the capacity to control the levels generated by LLMs using simple data augmentation and prompting.

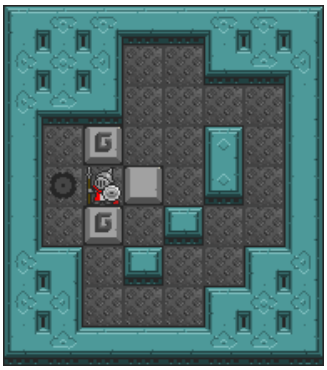


Figure 1: A level for the puzzle game *Sokoban* generated by GPT-3

Despite their impressive performance, there are reasons to doubt that LLMs would be well suited to the task of level generation. The first is representational. For context, the last few years have seen a steady increase in the use of machine learning to generate novel game content, including game levels. This procedural content generation through machine learning (PCGML) has made use of a variety of methods, including cellular automata, Markov models, convolutional neural networks, and generative adversarial networks. While dissimilar in function, these methods are nonetheless unified in that they tend to represent game levels spatially, as arrangements of tiles or features in two or three dimensions. This is an intuitive approach, as it allows models to more readily learn the local spatial dynamics present in game environments. By contrast, LLMs process inputs and generate outputs in a linear fashion. Game levels must be presented as a sequence of tokens in order to be fed into the model, and generated outputs must be reinterpreted as spatial data in order to be used. Further, variable-length tokenization schemes used by modern LLMs mean that two levels of the same size might

FDG 2023, April 12–14, 2023, Lisbon, Portugal

ACM ISBN 978-1-4503-9855-8/23/04…$15.00
https://doi.org/10.1145/3582437.3587211

be represented with different amounts of underlying tokens. Maintaining regularity and spatial relationships (e.g. attempting to place an enemy directly beneath a player in two dimensions) therefore requires more than simply counting the number of tokens. Nevertheless, prior work on n-grams and recurrent neural networks has demonstrated that game levels *can* be represented sequentially without the loss of critical spacial dependencies, albeit with some difficulty. We investigate the extent to which this holds for modern, attention-based models and their typical tokenization schemes.

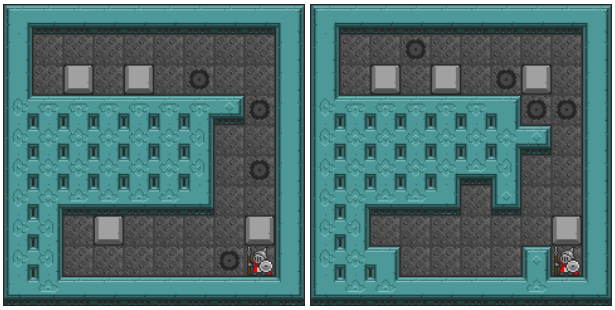
The second potential issue with using LLMs to generate game levels is that of data. LLMs are notoriously data hungry, while datasets of game levels are notoriously small, difficult to obtain, and lacking in standardization. The first obvious question is whether the ability of an LLM to generate levels depends on its receiving vast amounts of high-quality training data. More subtly, it is also important to determine whether the vast amount of data used in pretraining actually assists the LLM in producing game levels. It is not clear whether the patterns and structures learned from exposure natural language (or, in some cases, code) transfer to the functional and spatial constraints of game levels. The quality and even playability of a game level is often dependent on factors such as topology or the relative amounts of different tile types – a far cry from the syntax of English or Python!

Even so, LLMs also seem to have certain advantages when it comes to level generation, namely controllability and generalizability. Controllability here refers to the possibility of using natural language prompts to generate levels with particular characteristics. Recent work has demonstrated that natural language-guided generation is possible not only for text, but also for images [18] and music [1]. These systems leverage LLMs and are capable of accommodating a wide range of potential prompts, an impressive feat that provides some reason for optimism that current approaches for controllable level generation could be similarly improved. At the same time, LLMs have shown considerable promise in generalizing to unseen domains [2] or across a wide variety of tasks [19]. With respect to level generation, this might allow for a single LLM-based model to produce levels for multiple games or even, with sufficiently detailed prompting, a previously unencountered game.

In this paper, we aim to answer some of the initial questions surrounding the ability of LLMs to generate game levels using the iconic puzzle game *Sokoban*. We perform experiments on the effects of pretraining and dataset size, as well as a preliminary investigation on the controllability of LLM level generators. We conclude with a discussion of the results and the many fruitful avenues for future work.

2 RELATED WORK

Procedural content generation (PCG), refers to the use of automated or algorithmic methods to create artifacts, typically for use in art or games. Techniques for PCG range from simple noise functions to complex neural models. Our work falls into the broad category of procedural content generation via machine learning (PCGML) [29], in which content generating functions are learned from extant datasets. Liu et al. present an overview of the PCGML field, with a specific focus on deep learning [12].



(a) Playable level generated by GPT-2 **(b) Nearest level (edit distance) in the Boxoban set**

Figure 2: A novel and playable generated level, and its nearest neighbor in the training set.

For the specific task of generating game levels, common model choices include variational autoencoders [24], generative adversarial networks [16, 31], evolution [23], and reinforcement learning [10]. In addition, however, there is a history of using autoregressive models typically found in natural language processing for game level generation. Dahlskog et al. present early work on this approach, using a simple n-gram approach to generate novel Super Mario Bros. levels from an existing dataset by treating a level as a left-to-right sequence of "tokens" each representing a vertical slice [4]. This work was quickly expanded to use long short-term memory (LSTM) networks [28], a model choice which has also found success in generating levels based on human play traces [27] and combining levels from multiple games [20].

Our work also borrows from the literature on controllable PCG, in which specific parameters are provided to the generator in order to guide its outputs. Approaches for controllable PCG often involve manipulating a latent embedding vector, with prior work having made use of generative networks [14], VAEs [21, 22], and reinforcement learning [5].

Our target domain of *Sokoban* is a popular choice for PCG work, ranging from early rule- and template-based approaches [15, 30], to search-based methods [8] and recurrent neural networks [26]. Zakaria et al. present a particularly in-depth comparison of various PCG methods for *Sokoban*, including LSTMs [33]. They bootstrap an initially-small dataset of levels and use it to train their level generators, demonstrating that LSTMs are capable of producing a variety of novel and playable levels. They also perform experiments on the controllability of their generators. Their results indicate that while the task is challenging, LSTMs, in addition to other existing PCGML methods, can adhere to specified characteristics at levels substantially above chance. Our analysis is similar, though we instead focus our attention on a more modern class of large language model. In addition to a change in architecture, we also specifically interrogate some of the standard assumptions about the training and behavior of LLMs, and whether or not they are helpful for the task of level generation.

Finally, contemporaneous work makes use of a transformer-based language model to controllably generate levels for Super

Mario Bros [25]. This investigation is complementary to ours, and focuses on a different target domain.

3 DATA

We train our models on levels from the game *Sokoban*, a block-pushing puzzle game released in 1982 by Thinking Rabbit. In *Sokoban*, the player is tasked with navigating along a rectangular grid in order to push boxes into specified target squares. A level can easily be represented as a grid of ASCII characters, where each character is mapped to either: a wall, an empty space, the player, a box, a goal, a box on top of a goal, or a player on top of a goal. Despite its simplicity of representation, *Sokoban* levels can be very challenging for both human and artificial agents alike, owing to its rapidly-branching state space and the fact that certain game states are "unrecoverable" and, once reached, cannot be escaped from.

We use two sets of *Sokoban* levels to train our models. The first is the `Microban`[1] dataset, consisting of roughly 500 levels created by David W. Skinner. We restrict our dataset to 282 levels for which an ASTAR search agent was able to find a solution. Levels in this set range in size from 5 by 3 to 27 by 12, with solution lengths ranging from 1 to 279.

The second set of *Sokoban* levels is the `Boxoban` dataset, which consists of 438,000 procedurally generated levels. Levels were generated using a combination of heuristic and pattern-based rules [32]. Unlike with the `Microban` set, levels in the `Boxoban` set are all 10 by 10 and contain 4 boxes / goals. We use the ASTAR agent to recover solutions for all 438,000 levels, and solution lengths range from 6 to 206.

To construct our dataset, we split each level into a set of lines (applying a padding of walls in the case of non-rectangular levels), concatenate them to form a single string of characters, and finally apply the model's tokenizer.

4 MODELS

Our core experimental model is the Generative Pre-Trained Transformer (GPT), a class of attention-based language model [2, 17]. Both GPT-2 and GPT-3 are trained by attempting to predict the next "token" (typically a word or word piece) given the context of preceding tokens. Owing to its greater availability, we focus the majority of our experiments on GPT-2 and variants thereof.

5 METRICS

To measure the ability of LLMs to generate game levels, we use the following four metrics:

- *Playability*: we measure the proportion of generated game levels that are "valid". In the case of Sokoban, this means that they are rectangular, contain only valid characters, contain an equal and non-zero number of boxes and goals, contain exactly one player, and are solvable. We determine solvability using an ASTAR tree-search agent. If, after running for 150,000 steps, the ASTAR agent fails to find a valid solution, we deem the level unplayable. This provides a lower bound on the true rate of playability.
- *Novelty*: we measure the proportion of generated game levels that are distinct from each level in the training dataset. We use a simplified approach that treats two levels as distinct if their string edit distance is above some threshold. We note, however, that this definition of novelty does not take into account *functional* differences between levels (e.g. two levels may differ in only a single tile but nonetheless have very different solutions). Exploring the effects of different novelty measures remains an interesting area for future work. For our experiments, we use a the edit-distance approach and a threshold of $k = 5$.
- *Diversity*: we measure the proportion of generated game levels that are mutually distinct from each other. Using the same definition of distinctness as above, we use a graph-based approach to find the largest subset of generated levels that are all at least $k = 5$ edit distance from each other. Specifically, we convert the set of generated levels into an undirected graph where two nodes (levels) have an edge if their edit distance is above the threshold k. We then find the largest clique (subset of fully connected nodes) on this graph, and report the diversity as the size of this clique divided by the number of generated levels (a set of levels on this graph is only fully connected if each level is at least k edit distance away from every other level in the set). Because finding a maximum clique can be computationally intractable, we terminate the clique-finding algorithm after a specified number of iterations (1 million) and report the size of the largest clique found. This provides a lower bound on the model's diversity.
- *Accuracy*: in the case of controllability experiments, we measure the proportion of generated game levels that adhere to the given prompt. Rather than enforce an exact match between prompt and output, we allow the model to generate levels that are within a certain experiment-specific tolerance of the specified characteristic (for instance, with a tolerance of 5 we would consider accurate a level with a solution length of 21, when the prompt called for a level with solution length 25)

6 EXPERIMENTS

6.1 Effects of Pretraining

Our first experiment aims to answer two questions:

(1) Are LLMs capable of generating novel, playable game levels?
(2) Does the extensive pretraining given to LLMs affect their ability to generate game levels?

LLMs are typically trained on vast quantities of text collected from a variety of natural language contexts and then later "fine-tuned" on a smaller, more task-specific dataset. While pretraining has been shown to improve models' performance on a variety of downstream linguistic tasks, it is less clear whether it would help in the more specialized task of generating valid game levels. To examine this question, we consider 3 variants of the GPT-2 model: *standard*, *java-adapted*, and *untrained*. The *standard* GPT-2 model was pretrained on the `WebText` dataset, consisting of the content of 45 million links [17], the *java-adapted* model was pretrained on the `CodeSearchNet` dataset of Java code, consisting of 1.6 million Java

[1] Microban dataset available here: https://tinyurl.com/yckwxd7k

methods [13], and the *untrained* model, unsurprisingly, received no pretraining (weights are randomly initialized). As a note, both *standard* and *java-adapted* models use specialized tokenizers which are trained to efficiently break input strings into sub-word tokens. Rather than use an existing tokenizer, we allowed the *untrained* model to train a custom tokenizer on the game level dataset, using the same byte-pair encoding scheme as GPT-2.

Each model is trained for 100k steps with 5 random seeds on the Boxoban dataset with the following training hyperparameters: learning rate of 0.0001, weight decay of 0.0001, a batch size of 32, and the AdamW optimizer.Each model took roughly 24 hours to train on a single A100 GPU. In order to evaluate a model, we provide it with some initial context (for this experiment, only the START token) and then use beam search with random sampling in order to generate one or more continuations. We then compute the proportion of generated levels that are novel, the proportion that are playable, and the proportion that are novel, playable, and diverse. For simplicity, we call the proportion of novel, playable, and diverse levels the model's **score** (e.g. if the model produces 54 levels that are playable and novel out of 100 samples, of which 47 are mutually distinct, we report a score 0.47).

Because the outputs of a LLM are largely dependent on the hyperparameters used during generation, for each model and seed we perform an additional sweep over the generation *temperature*, the *top-p* value, and the number of *beams*. Each inference takes only a few minutes on a single A100 GPU. The entire sweep was completed in roughly 2 hours. For each model, we select the evaluation hyperparameters which achieve the highest score when averaged over the 5 random seeds. We report these average scores, along with the average novelty, playability, and diversity rates, for each model in Table 1.

6.2 Effects of Dataset Size

Another well-known property of LLMs is that their performance on a variety of NLP tasks tends to scale with the amount of training data [7], but does this trend hold for the specialized task of generating game levels? This question is particularly important because in many situations it is difficult or impossible to collect a large set of high-quality game levels. Relatedly, in situations where large amounts of game levels *are* available (typically games for which heuristic or rule-based PCG approaches exist), do LLMs benefit from ever-increasing dataset sizes? Finally, can simple data augmentation approaches improve LLM performance?

First, we consider four "slices" of the Boxoban dataset consisting of 0.1%, 1%, 10%, 100% (i.e. the complete dataset used above) of the data, randomly sampled. We take the standard GPT-2 model and re-train it on each of the slices for 100k steps, using the same training hyperparameters as above. We then evaluate each model's novelty, playability, and "score" using the same procedure as in Section 6.1. As before, we find the evaluation hyperparameters that achieve the highest average score for each model, and report the results in Table 2.

Next, we train a GPT-2 model on the Microban dataset, as well as two augmented versions of the dataset: $\texttt{Microban}_{\texttt{flip}}$ (levels flipped about the X and Y axes) and $\texttt{Microban}_{\texttt{flip+rotate}}$ (levels rotated 90 degrees clockwise and counterclockwise). Each model is trained for 100k steps, and the highest achieved average score for each is reported in Table 3.

Model	Novelty	Playability	Diversity	Score
GPT-2	**0.97**	0.54	**1.00**	0.53
GPT-2 (Untrained)	0.96	**0.60**	**1.00**	**0.56**
Java GPT-2	**0.97**	0.54	**1.00**	0.53

Table 1: Novelty, playability, diversity, and overall "score" (defined as the diversity of the subset of generated levels that are both novel and playable) for each type of pretraining, using the best evaluation hyperparameters when averaged over 5 seeds.

```
prop_empty: 0.25
solution_len: 65
    ########
    ##----##
    ##.-..##
    ###$-@-#
    #-$--$-#
    #---####
    ########
```

```
prop_empty: 0.269
solution_len: 42
    #########
    #---#####
    #---#---#
    ##-$*@--#
    ##-*.--##
    ##--#####
    #########
```

Figure 3: Two levels taken from the Microban dataset, along with their annotations

6.3 Controllability

Arguably the most compelling reason to use LLMs for game level generation is the ability to prompt the model in natural language to generate levels with specific characteristics. For instance, it might be possible to create a level that has a specific difficulty (represented by the length of the solution), or with certain level topologies. Recent LLMs have demonstrated impressive abilities to leverage prompting in order generalize from few or even zero examples on a variety of tasks [2]. However, zero-shot generalization is likely to be difficult for level generation owing to the many functional constraints on valid game levels and their dissimilarity from inputs encountered during pretraining. Thus, we instead focus on LLMs that have been trained specifically to adhere to prompts during level generation. We accomplish this by simply prepending an "annotation" to each level in the training dataset. Two examples of annotated levels are presented in Figure 3. At generation time, we provide the model with only the annotation and task it with generating the rest of the level while adhering to the specified values.

For this experiment, we focus on two annotated characteristics: the *proportion of empty space* (i.e. percentage of level tiles that are not players, walls, boxes, or targets) and the *solution length*. Both of these are measurable characteristics of valid *Sokoban* levels, though they differ in complexity. The proportion of empty space is an *observable* characteristic of a level, requiring only the ability to count in order to compute. Solution length, by contrast, can typically only be computed by actually solving the level in question

% of Boxoban	Novelty	Playability	Diversity	Score
0.1%	0.00	**0.80**	0.01	0.01
1%	0.10	0.66	0.97	0.03
10%	0.90	0.55	**1.00**	0.47
100%	**0.97**	0.54	**1.00**	**0.53**

Table 2: Novelty, playability, diveristy, and overall score for GPT-2 trained on increasing amounts of the Boxoban dataset, using the best hyperparameters averaged over 5 seeds. Increasing dataset size leads to increased performance.

Dataset	Novelty	Playability	Diversity	Score
Microban	**0.59**	0.30	0.83	0.02
$\text{Microban}_{\texttt{flip}}$	0.56	0.32	**0.89**	0.02
$\text{Microban}_{\texttt{flip+rotate}}$	0.24	**0.54**	0.82	**0.04**

Table 3: Novelty, playability, diversity, and overall score for GPT-2 trained on the Microban dataset and two augmentations, using the best hyperaparmeters averaged over 5 seeds. The model broadly overfits and fails to generate novel and playable levels.

and not through direct observation. Even visually sparse or simple levels can require long solutions.

As with the dataset size experiment in Section 6.2, we use a standard pretrained GPT-2 model. We train a separate model on the Boxoban dataset annotated with the proportion of empty space and the Boxoban dataset annotated with level solution length. At test time, we provide the model with only the annotation, randomly sampled from the collection of annotations in the training set. In addition to novelty, playability, and diversity, we compute the model's accuracy as described in Section 5. For the proportion of empty space condition, we use a tolerance of 0.01, and for the solution length condition we use a tolerance of 5. For this experiment, we report both the standard "score" defined above, as well as the "control score," which is simply the diversity of levels that are accurate to the prompt, in addition to being novel and playable. We report these results, using the same evaluation procedure as in Section 6.1, in Table 4.

6.4 Preliminary Investigation on GPT-3

While GPT-2 has demonstrated very high performance on a variety of natural language tasks, it has nonetheless been largely eclipsed by its successor: GPT-3, which boasts both substantially more parameters as well as a greatly increased amount of pretraining data. Access to GPT-3 is currently limited, making it infeasible to perform direct comparisons with GPT-2 on all measures. Nevertheless, we perform some initial experiments on the performance of OpenAI's *Davinci* model when trained on the Microban dataset and its augmentations.

We train the *Davinci* model for 10 epochs separately on each of the datasets using a single seed. At test time, we perform a limited hyperparameter sweep over generation temperature and top-p. As with GPT-2, we compute the model's novelty, playability, and overall score. We report the GPT-3 results in Table 5.

7 RESULTS

7.1 Effects of Pretraining

We see in Table 1 that all three models are able to generate novel, playable, and diverse levels. An average "score" of around 0.55 indicates that the language model is able to reliably generate *Sokoban* levels that are valid and solvable without directly copying from its Boxoban training dataset. We observe that the untrained GPT-2 model performs very slightly better than either of the pretrained models. The difference, however, is minute and likely to the effect of random variance. Overall, this seems to indicate that the pretraining afforded to these LLMs neither particularly helps nor hinders their ability to generate game levels. This could be explained by the substantial dissimilarity between modeling natural language and *Sokoban* levels, meaning that models are required to effectively learn from scratch in this domain and are able to do so.

7.2 Effects of Dataset Size

The results in Table 2 and Table 3 indicate that dataset size is indeed an important factor for an LLM's ability to generate game levels. For small datasets (i.e. the 0.1% and 1% conditions of Boxoban, as well as also Microban conditions), GPT-2 can produce levels that are independently novel or playable in isolation, but not levels that are both, leading to low overall scores. Nevertheless, in all but the 0.1% Boxoban condition, sample diversity remains relatively high. While the effect is not especially pronounced, there does appear to be a correlation between the size of the dataset and the model's score. This supports the notion that, like with many natural language tasks, LLM performance on level generation scales effectively with the availability of training data. We note, however, that prior works demonstrates LSTMs are capable of generating novel levels when trained on a bootstrapped dataset consisting originally of only 12 samples [33], meaning that it is unlikely that LLMs are fully *incapable* of performing well when restrcted to small datasets. What might account for this difference in performance, then? One possibility is expressivity: modern transformers are much better able to represent sequential data than LSTMs, and so are more likely to completely model the dynamics of their training datasets, to the detriment of their generative capabilities. However, as we will see, this explanation does not account for the performance of GPT-3 (see Section 7.4).

7.3 Controllability

In Table 4, we see effects of sampling levels conditioned on simple prompts. In the first row, we see that the GPT-2 model is able to produce levels that are novel, playable, and within a single tile of the specified proportion of empty space (corresponding to perfect accuracy and a relatively high control score). However, when it comes to solution length, GPT-2 achieves an accuracy of only 17%. Given the tolerance of 5 and the fact that most solution lengths in the dataset fall within a relatively narrow band, this cannot

Controls	Novelty	Playability	Accuracy	Diversity	Score	Control Score
Prop. Empty	**0.96**	0.57	**1.00**	0.97	**0.53**	**0.53**
Solution Len	0.95	0.54	0.17	**1.00**	0.50	0.14
Prop. Empty & Solution Len	**0.96**	**0.59**	0.03	0.79	0.45	0.03

Table 4: Novelty, playability, diversity, and accuracy (along with the overall score and the "control score", which accounts for accuracy) for GPT-2 trained on Boxoban, annotated with the proportion of empty space, the solution length, and both simultaneously, using the best hyperparameters when averaged over 5 seeds. The model is able to adhere to the empty space controls, but not the solution length controls.

Dataset	Novelty	Playability	Diversity	Score
Microban	0.09	0.88	0.67	0.01
$\texttt{Microban}_{\texttt{flip}}$	0.55	**0.94**	0.77	0.36
$\texttt{Microban}_{\texttt{flip+rotate}}$	**0.70**	0.93	**0.88**	**0.51**

Table 5: Novelty, playability, diversity, and overall score for GPT-3 trained on the Microban dataset and two augmentations. GPT-3 is able to produce novel, playable, and diverse levels from a relatively small training set.

be interpreted as anything more than the effects random chance. A similar fact holds for the combined condition, where overall accuracy is determined by both the correct amount of empty space and solution length and does not rise above 3%. It is worth noting, however, than even in the conditions where GPT-2 failed to produce accurate levels, it nonetheless continued to generally produce novel and playable ones. In other words, the introduction of the prompt did not negatively affect the model's performance.

7.4 Preliminary Investigation on GPT-3

Table 5 contains the results of GPT-3 level generation when trained on the Microban and its augmentations. While these results should be taken with a healthy amount of caution because they are generated from only a single training run and with a limited evaluation hyperparameter sweep, they nonetheless offer some reason for optimism. In contrast to GPT-2, GPT-3 is able to produce novel and playable levels when trained on both the augmented forms of the Microban dataset, with its overall score on the final condition approaching that of GPT-2 trained on the entire Boxoban dataset. As with previous experiments, however, we observe that increasing dataset size (in this case adding rotations in addition to flips) does lead to increased overall performance with GPT-3. In future work, we intend to perform a more robust analysis of GPT-3's abilities, including its capacity for controllable level generation.

8 FUTURE WORK

In this paper, we examine the performance of LLMs on generating levels for a single game. However, one of the primary strengths of LLMs is their ability to rapidly adapt to a variety of contexts given the appropriate prompt. Consider a dataset of levels from many different games, where each level has been annotated with the natural language mapping from tiles to game objects (e.g. "@ represents the player, M represents a monster), along with a description of the level objective. An LLM might be better equipped than other PCG systems to generate novel and playable levels from this variety of games, owing to its familiarity with natural language and capacity for rapid adaptation.

However, our work also indicates that making effective use of LLMs for game level generation may require more consideration of dataset size: few games have available the massive amount of levels present in the Boxoban set. As mentioned in Section 7.2, prior work has demonstrated that bootstrapping larger training sets from initially small collections of levels is a viable technique. Another possibility is augmenting existing datasets beyond simple flips and rotations. More generally, we should consider "fundamental tension of PCGML" [9]: at what point does the cost of obtaining training data for automated content generators exceed the cost of making the content by hand? While it's possible that LLMs require too much data to be feasible game content generators, the reasonable performance of GPT-3 on the small Microban dataset offers some optimism that this tension might be ameliorated by more sophisticated models.

It is also important to note that the large amounts of data used to pretrain LLMs could potentially include *Sokoban* levels in various formats. This fact complicates the notion of "novelty," as it's possible for the model to produce levels that are distinct from its fine-tuning dataset but are nonetheless copies of extant game levels. One potential approach for mitigating this danger would be to separate out the prompt encoding and level generation systems and use a model without pretraining for the latter (while still retaining the benefit of pretraining for the component of the model that understands natural language prompts).

Finally, there is room for much greater sophistication in the techniques used to control LLM outputs. Research in the area of controllable language model decoding [11] offers the opportunity to leverage existing work in PCG through reinforcement learning. More modern LLMs, especially, have also been shown to benefit from careful prompt engineering [34]. A combination of these approaches might allow for LLM generators that are better equipped to obey the functional constraints of game levels.

9 CONCLUSION

Large languages models are highly versatile. Beyond merely predicting likely continuations of text, they are capable of an impressive

range of natural language tasks. In this work, we show that generating video game levels can be added to that list. With sufficient data and training, LLMs are able to produce a diverse set of novel and playable *Sokoban* levels. We show that the pretraining generally afforded to these models does not hinder its ability to generate game levels, though any actual effect is unclear. We also demonstrate that, for GPT-2, the domain of game level generation is beholden to the same data scaling trends that apply to many natural language domains – model performance is strongly dependent on the availability of data. Cutting-edge LLMs like GPT-3 may have the potential to better generalize from small amounts of training data, though more work must be done before decisive conclusions can be drawn. With respect to controllability, we find that a simple prompting approach is sufficient for observable level characteristics like the proportion of empty tiles, but breaks down on more complicated metrics like solution length. Overall, the use of LLMs for game level generation shows promise despite it being a wildly different domain from natural language, complete with its own set of constraints and syntax. LLMs also seem potentially poised to overcome the general lack of available game level data, potentially offering a new way forward for procedural content generation through machine learning.

ETHICAL STATEMENT

Large language models have known biases and limitations, and can occasionally produce harmful or toxic text. While our models are trained to produce game levels, such training does not entirely eliminate this possibility. In addition, we note the possibility of LLMs copying published game levels included in their pretraining corpuses, a fact which should be considered before any form of widespread or commercial adoption.

REFERENCES

[1] Andrea Agostinelli, Timo I Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, et al. 2023. MusicLM: Generating Music From Text. *arXiv preprint arXiv:2301.11325* (2023).

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. https://doi.org/10.48550/ARXIV.2005.14165

[3] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems* 34 (2021), 15084–15097.

[4] Steve Dahlskog, Julian Togelius, and Mark J Nelson. 2014. Linear levels through n-grams. In *Proceedings of the 18th International Academic MindTrek Conference: Media Business, Management, Content & Services*. 200–206.

[5] Sam Earle, Maria Edwards, Ahmed Khalifa, Philip Bontrager, and Julian Togelius. 2021. Learning controllable content generators. In *2021 IEEE Conference on Games (CoG)*. IEEE, 1–9.

[6] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. 2018. Music transformer. *arXiv preprint arXiv:1809.04281* (2018).

[7] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361* (2020).

[8] Bilal Kartal, Nick Sohre, and Stephen J Guy. 2016. Data driven Sokoban puzzle generation with Monte Carlo tree search. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*.

[9] Isaac Karth and Adam M Smith. 2019. Addressing the fundamental tension of PCGML with discriminative learning. In *Proceedings of the 14th International Conference on the Foundations of Digital Games*. 1–9.

[10] Ahmed Khalifa, Philip Bontrager, Sam Earle, and Julian Togelius. 2020. Pcgrl: Procedural content generation via reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, Vol. 16. 95–101.

[11] Jiwei Li, Will Monroe, and Dan Jurafsky. 2017. Learning to decode for future success. *arXiv preprint arXiv:1701.06549* (2017).

[12] Jialin Liu, Sam Snodgrass, Ahmed Khalifa, Sebastian Risi, Georgios N Yannakakis, and Julian Togelius. 2021. Deep learning for procedural content generation. *Neural Computing and Applications* 33, 1 (2021), 19–37.

[13] Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, Alexey Svyatkovskiy, Ambrosio Blanco, Colin Clement, Dawn Drain, Daxin Jiang, Duyu Tang, et al. 2021. Codexglue: A machine learning benchmark dataset for code understanding and generation. *arXiv preprint arXiv:2102.04664* (2021).

[14] Justin Mott, Saujas Nandi, and Luke Zeller. 2019. Controllable and coherent level generation: A two-pronged approach. In *Experimental AI in games workshop*.

[15] Yoshio Murase, Hitoshi Matsubara, and Yuzuru Hiraga. 1996. Automatic making of sokoban problems. In *PRICAI'96: Topics in Artificial Intelligence: 4th Pacific Rim International Conference on Artificial Intelligence Cairns, Australia, August 26–30, 1996 Proceedings 4*. Springer, 592–600.

[16] Kyungjin Park, Bradford W Mott, Wookhee Min, Kristy Elizabeth Boyer, Eric N Wiebe, and James C Lester. 2019. Generating educational game levels with multistep deep convolutional generative adversarial networks. In *2019 IEEE Conference on Games (CoG)*. IEEE, 1–8.

[17] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[18] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125* (2022).

[19] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. 2022. A generalist agent. *arXiv preprint arXiv:2205.06175* (2022).

[20] Anurag Sarkar and Seth Cooper. 2018. Blending Levels from Different Games using LSTMs.. In *AIIDE Workshops*.

[21] Anurag Sarkar and Seth Cooper. 2021. Dungeon and platformer level blending and generation using conditional vaes. In *2021 IEEE Conference on Games (CoG)*. IEEE, 1–8.

[22] Anurag Sarkar, Zhihan Yang, and Seth Cooper. 2020. Conditional level generation and game blending. *arXiv preprint arXiv:2010.07735* (2020).

[23] Jacob Schrum, Jake Gutierrez, Vanessa Volz, Jialin Liu, Simon Lucas, and Sebastian Risi. 2020. Interactive evolution and exploration within latent level-design space of generative adversarial networks. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. 148–156.

[24] Sam Snodgrass and Anurag Sarkar. 2020. Multi-domain level generation and blending with sketches via example-driven bsp and variational autoencoders. In *Proceedings of the 15th international conference on the foundations of digital games*. 1–11.

[25] Shyam Sudhakaran, Miguel González-Duque, Claire Glanois, Matthias Freiberger, Elias Najarro, and Sebastian Risi. 2023. MarioGPT: Open-Ended Text2Level Generation through Large Language Models. *arXiv preprint arXiv:2302.05981* (2023).

[26] Muhammad Suleman, Farrukh Hasan Syed, Tahir Q Syed, Saqib Arfeen, Sadaf I Behlim, and Behroz Mirza. 2017. Generation of sokoban stages using recurrent neural networks. *International Journal of Advanced Computer Science and Applications* 8, 3 (2017).

[27] Adam Summerville, Matthew Guzdial, Michael Mateas, and Mark O Riedl. 2016. Learning player tailored content from observation: Platformer level generation from video traces using lstms. In *Twelfth artificial intelligence and interactive digital entertainment conference*.

[28] Adam Summerville and Michael Mateas. 2016. Super Mario as a String: Platformer Level Generation Via LSTMs. https://doi.org/10.48550/ARXIV.1603.00930

[29] Adam Summerville, Sam Snodgrass, Matthew Guzdial, Christoffer Holmgård, Amy K Hoover, Aaron Isaksen, Andy Nealen, and Julian Togelius. 2018. Procedural content generation via machine learning (PCGML). *IEEE Transactions on Games* 10, 3 (2018), 257–270.

[30] Joshua Taylor and Ian Parberry. 2011. Procedural generation of sokoban levels. In *Proceedings of the International North American Conference on Intelligent Games and Simulation*. 5–12.

[31] Ruben Rodriguez Torrado, Ahmed Khalifa, Michael Cerny Green, Niels Justesen, Sebastian Risi, and Julian Togelius. 2020. Bootstrapping conditional gans for video game level generation. In *2020 IEEE Conference on Games (CoG)*. IEEE, 41–48.

[32] Théophane Weber, Sébastien Racanière, David P. Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adria Puigdomènech Badia, Oriol Vinyals, Nicolas

Heess, Yujia Li, Razvan Pascanu, Peter Battaglia, Demis Hassabis, David Silver, and Daan Wierstra. 2017. Imagination-Augmented Agents for Deep Reinforcement Learning. https://doi.org/10.48550/ARXIV.1707.06203

[33] Yahia Zakaria, Magda Fayek, and Mayada Hadhoud. 2022. Procedural Level Generation for Sokoban via Deep Learning: An Experimental Study. *IEEE Transactions on Games* (2022), 1–1. https://doi.org/10.1109/TG.2022.3175795

[34] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*. PMLR, 12697–12706.