Prompt Wrangling: On Replication and Generalization in Large Language Models for PCG Levels

Arash Moradi Karkaj
New Jersey Institute of Technology
Newark, New Jersey, USA
am3493@njit.edu

Mark J. Nelson
American University
Washington D.C., USA
mnelson@american.edu

Ioannis Koutis
New Jersey Institute of Technology
Newark, New Jersey, USA
i.koutis@njit.edu

Amy K. Hoover
New Jersey Institute of Technology
Newark, New Jersey, USA
ahoover@njit.edu

ABSTRACT

The ChatGPT4PCG competition calls for participants to submit inputs to ChatGPT or prompts that guide its output toward instructions to generate levels as sequences of Tetris-like block drops. Prompts submitted to the competition are queried by ChatGPT to generate levels that resemble letters of the English alphabet. Levels are evaluated based on their similarity to the target letter and physical stability in the game engine. This provides a quantitative evaluation setting for prompt-based procedural content generation (PCG), an approach that has been gaining popularity in PCG, as in other areas of generative AI. This paper focuses on replicating and generalizing the competition results. The replication experiments in the paper first aim to test whether the number of responses gathered from ChatGPT is sufficient to account for the stochasticity. We requery the original prompt submissions and rerun the original scripts from the competition, on different machines, about six months after the competition. We find that results largely replicate, except that two of the 15 submissions do much better in our replication, for reasons we can only partly determine. When it comes to generalization, we notice that the top-performing prompt has instructions for all 26 target levels hardcoded, which is at odds with the PCGML goal of generating new, previously unseen content from examples. We perform experiments in more restricted zero-shot and few-shot prompting scenarios, and find that generalization remains a challenge for current approaches.

CCS CONCEPTS

• **Computing methodologies** → *Natural language generation.*

KEYWORDS

Procedural content generation (PCG), Large Language Models (LLMs), Generalizability, Evaluating Generalization, Science Birds

FDG 2024, May 21–24, 2024, Worcester, MA, USA

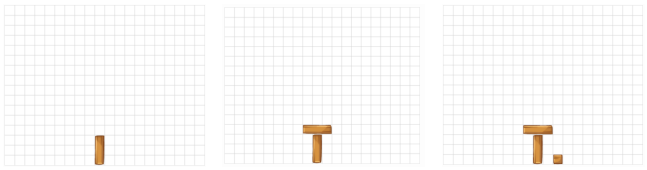
ACM ISBN 979-8-4007-0955-5/24/05.
https://doi.org/10.1145/3649921.3659853

ACM Reference Format:
Arash Moradi Karkaj, Mark J. Nelson, Ioannis Koutis, and Amy K. Hoover. 2024. Prompt Wrangling: On Replication and Generalization in Large Language Models for PCG Levels. In *Proceedings of the 19th International Conference on the Foundations of Digital Games (FDG 2024), May 21–24, 2024, Worcester, MA, USA*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3649921.3659853

1 INTRODUCTION

Given the recent success of large language models (LLMs) for generating natural language text [10], vision-language models for generating images [11, 12] and 3D meshes, audio-text models for generating audio [5], even disciplines like evolutionary computation [6, 7] and optimization more broadly [21] are reconceptualizing existing paradigms to include pre-trained generative models. Central to the recent popularity of generative AI are the affordances provided by *prompting*, where *prompts* are strings of natural language text input to a generative model. By simply explaining to the model in plain language what it should create, users are able to control the characteristics of the content that they generate. While some approaches to prompt-based procedural content generation (PCG) exist [15], the ChatGPT4PCG competition [17] held at the 2023 IEEE Conference on Games is the first approach to in prompt-generated PCG to foreground the affordances provided by prompting.

The ChatGPT4PCG competition [17] calls for participants to submit prompts to ChatGPT-3.5 that then responds with instructions to generate levels for the Science Birds physics-based game environment. These instructions are a series of commands to drop blocks at specific locations in the game, and ChatGPT-3.5 responses become XML levels through the architecture provided by the competition [1]. The goal is for a single prompt to generate levels that in aggregate resemble all 26 letters in the English alphabet. A single prompt generates levels that look like different letters through the template imposed by the competition design. The *object variable* in the prompt tells ChatGPT-3.5 which letter its instructions should generate (i.e., the *target letter*). Prompts are evaluated based on the performance of ten generated levels for each of the 26 letters in the alphabet. Assuming that a prompt generates ten valid levels per letter, it is evaluated on a total of 260 levels. The scores for the prompts are based on their resemblance to the target English letters and on their in-game physical stability. In a sense the competition



(a) Step 1: One-by-Three Block **(b) Step 2: Three-by-One Block** **(c) Step 3: One-by-One Block**

Figure 1: Three block drops.

empowers its participants to prompt-engineering to a level generator within the boundaries of parameterization imposed by the the competition's organizers [13, 19, 20].

Two sets of experiments are proposed. The **replication experiment** investigates whether the original prompts achieve approximately the same scores six months after the competition. ChatGPT-3.5 is closed source and it is unclear when changes are made that may impact the generated levels. This experiment also examines whether scores are impacted by increasing the number of levels generated per letter to 100. The results of the replication experiment show that ChatGPT-3.5 is changing over the time span of six months, and that these small changes can significantly impact how prompts score, whether ten or 100 levels are evaluated. While the sample size is likely adequate for the competition, to make conclusions about prompts submitted to the competition over the course of years, it will be important to query LLMs with static weights.

The **generalization experiment** examines the ability of prompts to generate levels it has not yet seen (i.e., that are not hard-coded in the prompt). Prompts are modified to contain zero-shots or example letters in the zero-shot setting or one-shot in the one-shot setting. The prompts in the one-shot setting are tested per letter given that prompts contain different example letters and some letters are more difficult to generate than others. N-shot prompting is specified for a small subset of prompts and example letters. The goal of this experiment is to understand whether successful prompts are *generalizing* or if it is the quality of their lookup tables that determines their score. Results show that these modified prompts perform significantly worse than their unmodified counterparts and prompts that generalize well are not necessarily those that perform well in the competition. These results suggest that generalizing with these new level generators is an open problem to explore.

Given that results are likely replicable with a static LLM, the generalization experiment in particular is motivated by the goal of aligning this new prompt-based PCG paradigm with the body of work related to procedural content generation through machine learning (PCGML), which focuses on generalizing from input examples to new, unseen generated content [16]. Judging by the results, it is worth looking beyond the original leaderboard for promising prompting strategies.

2 THE CHATGPT4PCG COMPETITION

ChatGPT for Procedural Content Generation (ChatGPT4PCG) is a new competition for generating game levels with LLMs. Level generators assemble levels by giving a series of instructions for where to drop differently shaped bricks. The physics simulation is provided by Science Birds, a Unity-based environment for game research (Science Birds also has other components, based on the popular game *Angry Birds*, but only the physics engine is used in this competition, not the bird-based gameplay) [3].

Prompt	Len.	Chars.	Token	Qualifies
AdrienTeam	Pass	Pass	Pass	Yes
albatross	Pass	Pass	Pass	Yes
Back to the future	Pass	Pass	Pass	Yes
BirdBirdBird	Fail	Pass	Pass	No
BirdsTeam	Fail	Pass	Pass	No
dereventsolve	Pass	Pass	Pass	Yes
For500	Pass	Pass	Pass	Yes
hachi	Pass	Pass	Pass	Yes
Harry Single Group	Pass	Pass	Pass	Yes
Hope	Pass	Pass	Pass	Yes
JUSTIN	Pass	Pass	Pass	Yes
pixelArt	Pass	Fail	Pass	No
Prompt_Wranglers	Pass	Pass	Pass	Yes
Saltyfish1884	Pass	Pass	Pass	Yes
Soda	Pass	Pass	Pass	Yes
Team Staciiaz	Pass	Pass	Pass	Yes
The Organizer	Pass	Pass	Pass	Yes
zeilde	Pass	Pass	Pass	Yes

Table 1: Qualification Conditions. Entries to the ChatGPT4PCG competition qualify by satisfying the following contraints: 1) contain 900 words or fewer, 2) contain only the types of characters permitted by the organizers, 3) contain an object token [17]. Three of the 18 prompts were disqualified, leaving 15 to be evaluated: 'BirdBirdBird' and 'BirdsTeam' exceed the maximum length, and 'pixelart' contains a forbidden character.

An example of building the letter 'T' is shown in Figure 1. The generator first drops a one-by-three brick at $x = 9$ (Figure 1a), then drops a three-by-one brick at $x = 9$ (Figure 1b), which stacks on the previous brick. Generators drop these blocks through the drop function *ab_drop*(*block_type*, *position*), where the first argument is one of the three available block shapes, and the second argument is the x-position of the drop. Besides the two bricks used to construct this 'T' shape, the third brick available is a one-by-one square, shown dropped next to the 'T' in (Figure 1c).

For participants, the challenge is to design a prompt template of 900 or fewer words that contains enough domain-specific information for ChatGPT to generate (from that prompt) twenty-six different levels, with each level resembling one of the twenty-six capital letters of the English alphabet.

These submitted prompt templates must contain a variable called `<OBJECT>`. This template variable is replaced by the actual letter to be generated before sending the prompt to ChatGPT. Because its default temperature is nonzero, ChatGPT often generates a different set of level instructions for identical prompts. Therefore to evaluate the submitted prompts, ChatGPT4PCG determines the quality of prompts by generating and evaluating ten levels per letter for a total of 260 levels per submitted prompt.

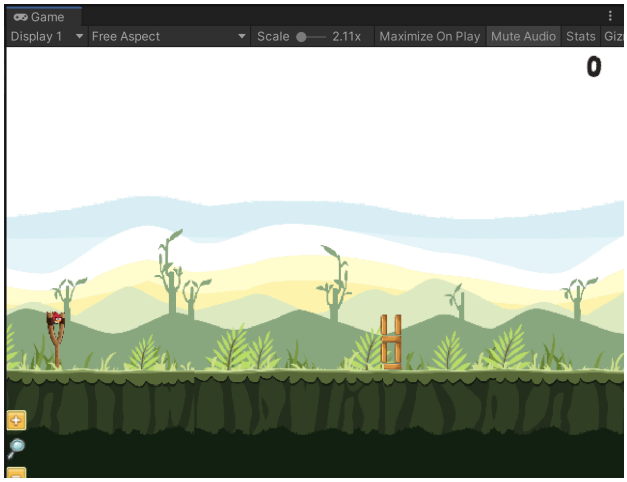


Figure 2: Example of a Prompt-Generated Level for Target Letter 'A.' Levels are generated in XML and loaded into the Unity game, Science Birds. The competition restricts level generation such that differences between levels are the number, position, and types of bricks in the list of GameObjects.

2.1 Prompt Qualification

Before generating levels, prompts are first assessed for eligibility. Any entry to the competition must conform to rules that restrict the maximum length of the prompt and the types of characters it contains [17]. Prompts must also contain the <OBJECT> token somewhere. Table 1 shows the results of the length, character type, and object token tests on the 18 entries [1] to the ChatGPT4PCG competition. Qualifying requires that each prompt passes each test. Note that while the score for the *v1_sample* prompt is recorded in the results available at https://chatgpt4pcg.github.io/2023/result, it is not available for download.

2.2 Level Generation

After qualifying, a prompt is input to ChatGPT $n = 10$ times per target letter for all 26 letters. The <OBJECT> variable in the prompt template is replaced with the specific target letter for each trial. A total of 260 responses are therefore generated per submission. An example ChatGPT response is shown in Figure 3a. The code between the triple back-ticks in the response is then extracted into an intermediate format, the code shown in Figure 3b. Although the x-position of the block is determined by the generated-code, the XML contains a y-value corresponding to where the block would sit in the game environment shown in Science Birds in Figure 2. If a level contains blocks that would exceed the width or height restrictions of the game, the XML for that level is not generated.

2.3 Level Evaluation

Levels are evaluated through a custom interface in Unity and assigned scores through a complex weighting scheme described by Taveekitworachai et al. [17]. The weighting is applied to the stability and similarity scores and is designed to assign high weights

[1]https://chatgpt4pcg.github.io/2023/files/prompts.zip

````
Task: Generate the uppercase English letter
'A'.
```
ab_drop('b31', 2)
ab_drop('b13', 1)
ab_drop('b13', 3)
ab_drop('b31', 2)
ab_drop('b13', 1)
ab_drop('b13', 3)
```
````

(a) Example Level 'A': ChatGPT Response

```
ab_drop('b31', 2)
ab_drop('b13', 1)
ab_drop('b13', 3)
ab_drop('b31', 2)
ab_drop('b13', 1)
ab_drop('b13', 3)
```

(b) Example Level 'A': Code Extracted from ChatGPT Response

Figure 3: ChatGPT Response for the Letter 'A'

to letters that are hard to generate. The details of the weighting scheme are left to Taveekitworachai et al. [17] and are omitted from these calculations for readability.

Measuring Stability: The stability test monitors a level for ten seconds in Unity and assign a score based on the number of blocks that fall. Equation 1 shows the stability of level x as its number of blocks total blocks minus the number of blocks that change position moving blocks. Because the number of blocks is likely to vary between levels, the stability score is this difference normalized by total blocks. Levels score a stability of 1.0 when moving blocks$(x) = 0$ and stability of 0.0 when moving blocks$(x) =$ total blocks(x).

$$\text{stability}(x) = \frac{\text{total blocks}(x) - \text{moving blocks}(x)}{\text{total blocks}(x)} \quad (1)$$

Measuring Similarity: The similarity test takes an in-game screenshot of a level ten seconds after loading it, the same amount of time allotted for calculating stability. The screenshot is then input to a fine-tuned Vision Transformer [2] trained to recognize handwritten letters. [2]. Activating the network on the image, the similarity score is softmax probability output by the node corresponding to the target letter. A similarity of 1.0 occurs when the softmax probability outputs 1 for the target node and 0.0 for all other output nodes. A similarity of 0.0 occurs when the softmax probability outputs 0.0 for the target node.

Calculating Scores:

An individual level is assessed by multiplying its scores for similarity and stability such that a similarity or stability score of 0 results in a 0 for the entire level:

[2]The model is available at https://huggingface.co/datasets/pittawat/letter_recognition and is trained on data from https://www.nist.gov/srd/nist-special-database-19

$$\text{levelscore}(x) = \text{stability}(x) \times \text{similarity}(x) \quad (2)$$

A letter score is then the average level score of the levels generated for that target letter. Let A represent the target letter and n levels (i.e., $x_1, x_2, ..., x_n$) are generated, then the level score for A is:

$$\text{letterscore}_A(x_1, x_2, ...x_n) = \frac{\sum_{i=1}^{i=n} \text{levelscore}(\text{x}_\text{i})}{n} \quad (3)$$

Then prompt scores are the average score per target letter:

$$\text{promptscore} = \frac{\text{letterscore}_A + \text{letterscore}_B + \ldots + \text{letterscore}_Z}{26} \quad (4)$$

3 EXPERIMENTS

Two sets of experiments are conducted. The replication experiment examines whether the changes that OpenAI makes to ChatGPT-3.5 during the span of six months can have a significant impact on prompt scores and whether any potential differences in scores can be explained through the number of responses per target letter. For the generalization experiment, original prompts are modified to remove instructions for specific letters, depending on generalization setting. For the Zero-Shot setting, all examples are removed from the original prompts. The One-Shot setting tests one-shot generalization from a single example letter. Only prompts that contain this letter are tested for one-shot generalization, and the example letter is varied from 'A' to 'Z.' The N-Shot setting looks at the performance of only two prompts, but tests whether generalization improves with the number of shots provided. All of the modified prompts and the original prompts are available at gaimes-njit.github.io.

3.1 Replication Experiment

Replicating and reproducing research results is a cornerstone of the scientific process [9], and our first step is to re-run the original competition and analyze the replicability of the results. This experiment first tests whether a sample size of $n = 10$ ChatGPT responses per letter per prompt (i.e., 260 responses in total) is large enough to assess the quality of the prompts. To test this parameter setting, the replication experiment runs the competition pipeline for $n = 10$ and for $n = 100$, scoring and ranking prompts based on a total of 260 and 2600 responses from ChatGPT respectively. If the ranks and scores do not change significantly between $n = 10$ and $n = 100$, we can conclude that $n = 10$ is sufficient.

In addition to the more narrow question of sample size, replicating the competition results tests the entire pipeline: whether ChatGPT produces the same responses now as it did a few months ago, whether the scripts run the same in our computational environment as on the organizers' machines, and so on. Section 4.1 presents the original results alongside our replicated results, and analyzes each of these factors.

3.2 Generalization Experiment

The **generalization experiment** explores whether prompts can *generalize* their level generation to create levels resembling unseen letters of the alphabet. The parameters varied are the example letters provided in each prompt, and as a result, the amount of generalization the LLM-based procedural content generator is expected to perform. These experiments are intended, in particular, to tie the ChatGPT4PCG competition in more directly to discussions around few-shot prompting in the LLM literature. To this end, three settings for the generalization experiment are proposed: Zero-Shot Generalization, One-Shot Generalization, and N-Shot Generalization.

Zero-Shot Generalization tests generalization when prompts explain how levels should be generated without referencing example solutions. The original competition prompts are modified to exclude any example instructions for letters. The zero-shot generalization test is a strong test of generalization, and it is a common way to test generalization in the LLM evaluation literature. In the ChatGPT4PCG case, zero-shot prompting means building a prompt that can include any amount of general discussion, but excludes explicit drop sequences for any letters.

One-Shot Generalization is a common way to prompt LLMs that are expected to generalize from one example solution. It is a strong test of generalization, but weaker than the zero-shot setting which is expected to generalize to other cases without an example solution. In this experiment, 26 modified versions of all competition entries are tested, one for each possible choice of one-shot letter. We modified the original prompts to remove any explicit solutions other than the one letter being tested each time. While some submissions have an example for each of the letters (namely The Organizer and dereventsolve), the rest include partial examples, so in some cases this results in the modified prompts actually being zero-shot. In those cases we excluded them from the results presentation.

N-Shot Generalization is based on the modified versions of 'The Organizer' and 'Prompt_Wranglers'. This experiment varies the number of examples ("shots") given, from zero (as in the zero-shot prompting experiment) all the way up to 26 (as in the top two entries in the original competition). The experiment intends to investigate generalization along a continuum from zero example solutions provided in the prompt to all 26 example letters. The experiment starts by removing all shots from the prompts, and slowly reintroducing examples for the letter 'A,' then 'A' and 'B', iteratively adding letters until all 26 shots are provided in each prompt.

4 RESULTS

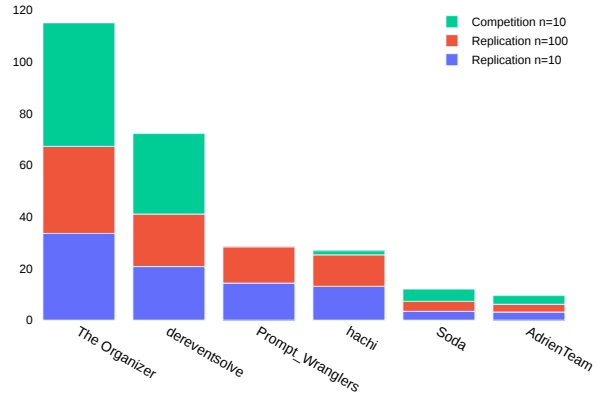
The results from the replication experiment show that the competition parameter corresponding to the number of responses collected from ChatGPT is sufficient to rank prompts reliably. The results from the generalization experiment show that generalization is still an open problem.

4.1 Replication Results

The left side of Table 2 shows the ranking of scoring of competition entries run for $n = 10$ and $n = 100$ in the replication experiment. The prompt names are listed in the middle and on the right are how these prompts rank and score in the competition. The replication results show similar scores and identical rankings for $n = 10$ and $n = 100$, indicating that $n = 10$ generates enough ChatGPT responses per letter to account for the stochasticity of the domain.

Table 2: Results from the Replication Experiment.

Replication			Teams	Competition	
Rank	n = 10	n = 100		Rank	n=10
1	33.58	33.71	**The Organizer**	1	47.84
2	20.75	20.34	dereventsolve	2	31.15
3	14.34	14.06	Prompt_Wranglers	14	0.00
4	13.11	12.18	hachi	9	1.57
5	3.42	3.87	Soda	3	4.76
6	3.14	3.02	AdrienTeam	4	3.35
7	2.34	2.09	Harry Single Group	8	1.86
8	2.12	1.96	Back to the future	10	1.38
9	2.09	1.90	Saltyfish1884	5	2.12
10	1.88	1.62	zeilde	6	2.12
11	1.81	1.42	Team Staciiaz	7	1.96
12	1.21	0.88	JUSTIN	11	0.52
13	0.20	0.23	albatross	13	0.02
14	0.00	0.00	Hope	12	0.15
15	0.00	0.00	For500	15	0.00



(a)

Figure 4: Scores by prompt. The scores for the competition and replication results for each prompt are stacked on top of each other.

Interestingly, there are significant differences in ranking between the replication results and the results of the competition. 'Soda' and 'AdrienTeam' rank third and fourth in the competition but drop to ranks five and six in the replication results. On the other hand, 'Prompt_Wranglers' increases from rank 14 to rank 3, and 'hachi' increases from rank 9 to 4.

Digging into individual prompts' replication performance, Figure 4 indicates that while 'Prompt_Wranglers' and 'hachi' score significantly better in the replication results than they did in the competition, 'The Organizer,' 'dereventsolve,' 'Soda,' and 'AdrienTeam' score more proportionately. Figure 5a shows the distribution of scores for the replication results and the competition results that includes the scores for 'Prompt_Wranglers' and 'hachi'. These replication results have higher interquartile ranges (IQRs) of 6.76 and 6.88 than that of the competition results with an IQR of 2.40. Removing 'Prompt_Wranglers' and 'hachi' from the three distributions in Figure 5b shows IQRs for the replication results (i.e., 1.93 and 2.14 for $n = 10$ and $n = 100$) much closer to the IQR of the competition, 2.83. These data suggest that the improvement of 'Prompt_Wranglers' and 'hachi' is largely responsible for the increased IQR in Figure 5a.

Given that $n = 10$ is a large enough sample size to account for stochasticty in the domain, the results suggest that the differences in ranking are caused by a difference in the responses generated by ChatGPT. In fact by generating level XML files from raw responses rather (i.e., skipping the intermediate stage that extracts code from the responses), 'Prompt_Wranglers' increases its score by 22.22 points. The code extraction step examines ChatGPT responses for a set of triple backticks and exclusively extracts the code contained between them. These triple backticks were absent in responses from ChatGPT when collected for the competition, but present in responses collected for the replication experiments. LLMs can be sensitive to the exact wording in a prompt, and responses to a prompt are likely to vary between LLMs and even between different versions of the same LLM [14].

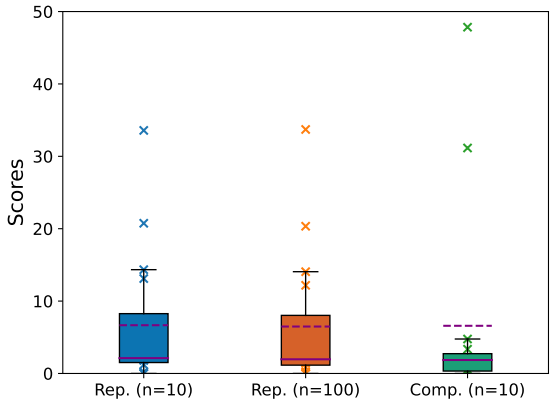
Table 3: Results from Zero-Shot Generalization.

Zero-Shot		Teams	Orig. Replication	
Rank	Raw Score		Rank	Raw Score
1	0.074	Soda	5	0.088
2	0.056	Team Staciiaz	11	0.043
3	0.054	Prompt_Wranglers	3	0.289
4	0.048	Harry Single Group	7	0.050
5	0.047	Saltyfish1884	9	0.052
6	0.046	Back to the future	8	0.050
7	0.038	zeilde	10	0.038
8	0.035	AdrienTeam	6	0.075
9	0.028	hachi	4	0.260
10	0.025	Hope	14	0.000
11	0.021	JUSTIN	12	0.034
=12	0.000	For500	15	0.000
=12	0.000	albatross	13	0.059
=12	0.000	dereventsolve	2	0.395
=12	0.000	The Organizer	1	0.605

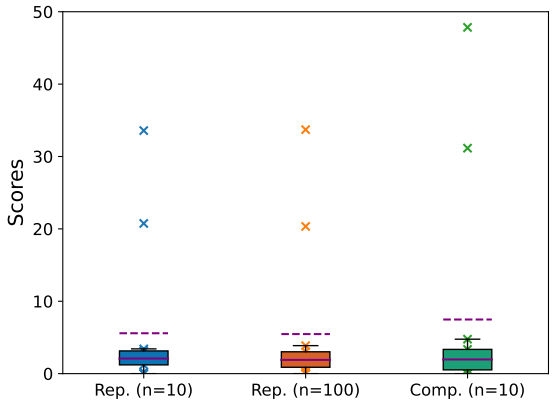
Overall the results suggest that a sample size of $n = 10$ responses is sufficient to rank prompts reliably but caution that the results of the competition and therefore quality of the prompts are largely dependent on the specific LLM that is queried.

4.2 Generalization Results

Table 3 and Figures 6 and 7 show the results of the generalization experiment. An important difference in scoring for the generalization experiment results is that all reported scores are *raw scores*. The competition's scoring process includes two "dynamic" aspects – weighting and normalization – that make scores incomparable



(a) Score Distributions Including 'Prompt_Wranglers' and 'hachi.'



(b) Score Distributions Excluding 'Prompt_Wranglers' and 'hachi.'

Figure 5: Score Distributions for the Replication and Competition Results. For each distribution, the outliers are scores for 'The Organizer' and 'dereventsolve.'

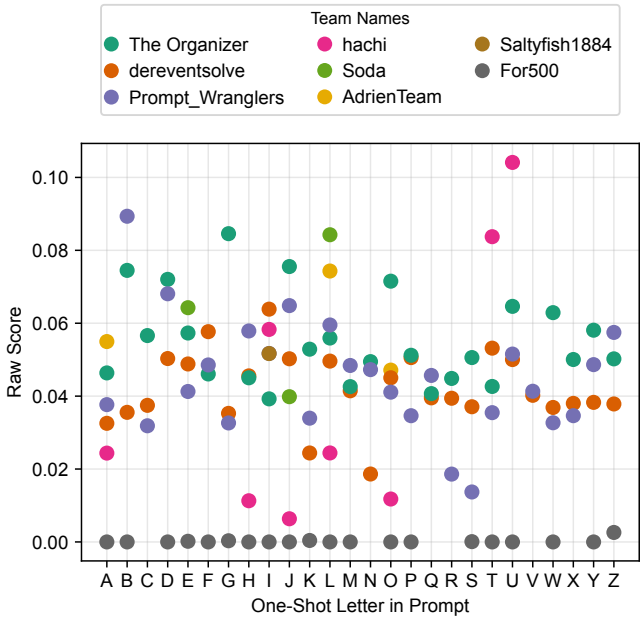


Figure 6: Results of the One-Shot Generalization Experiment. A subset of prompts include instructions for generating specific letters, and the performance for a prompt is shown only for prompts that contain the given shot-letter.

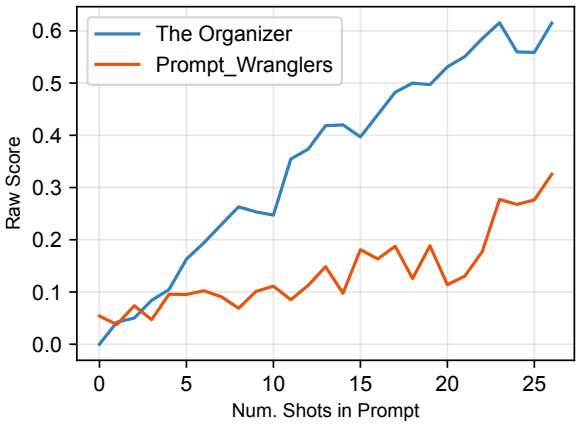


Figure 7: Results of the N-Shot Generalization Experiment. The number of shots N ranges from 0 at the left to 26 at the right. These experiments test two prompts: 'The Organizer' and 'Prompt_Wranglers'.

across experiments (or different iterations of the competition). Per-letter scores are weighted to give more weight to harder letters and less weight to easier letters (easier/harder measured by how well prompts do). The weighted scores are then normalized by dividing each prompt's score by the sum of all prompt scores and multiplying by 100, effectively making the final score a percentage (all scores in the competition total to 100). To make scores comparable between different experiments, a new metric called *raw score* is introduced. It removes the dynamic aspects of the original competition: letters are equally weighted, and there is no normalization (summarized in Section 2.3).

Zero-Shot Generalization Results: When restricted to the zero-shot setting, most prompts' performance is reduced significantly, as can be seen by comparing the two raw-score columns in Table 3. The top-performing zero-shot prompt has performance only about 12% of the top-performing prompt in the original competition (as replicated in our replication experiments). The rankings are also quite different, suggesting some prompts that didn't rank highly in the original competition, such as 'Soda', may be on closer to the right track nonetheless if our interest in is generalization.

One-Shot Generalization Results: In the one-shot setting, the best performing prompts improve up only slightly from the zero-shot results, as shown in Figure 6. Raw scores remain significantly below the scores seen for the best original prompts. However the performance of some individual prompts does improve significantly from the zero-shot setting. Overall performance varies quite a bit, and in some cases is highly dependent on *which* single example is given as the 1-shot example. For example, the single highest performing 1-shot prompt is 'hachi' with 'U' given as the one shot; 'hachi' also stands out when given 'T'. But this prompt doesn't perform exceptionally for any of the other 1-shot letters.

Overall prompts do not perform as well in the zero-shot or one-shot setting as they do in the unrestricted competition. This result suggests that strong generalization remains an open problem.

N-Shot Generalization Results: Figure 7 shows how performance varies, for two starting prompts, when the number of "shots" in the prompts varies from 0 (no explicit solution examples) to 26 (solution examples for all 26 letters). As expected, there is a general trend of increasing performance with increasing shots: as additional explicit solutions are given in the prompt, performance improves. The relatively linear increases in performance also suggest that not much generalization is going on: providing twice as many explicit solutions approximately doubles performance.

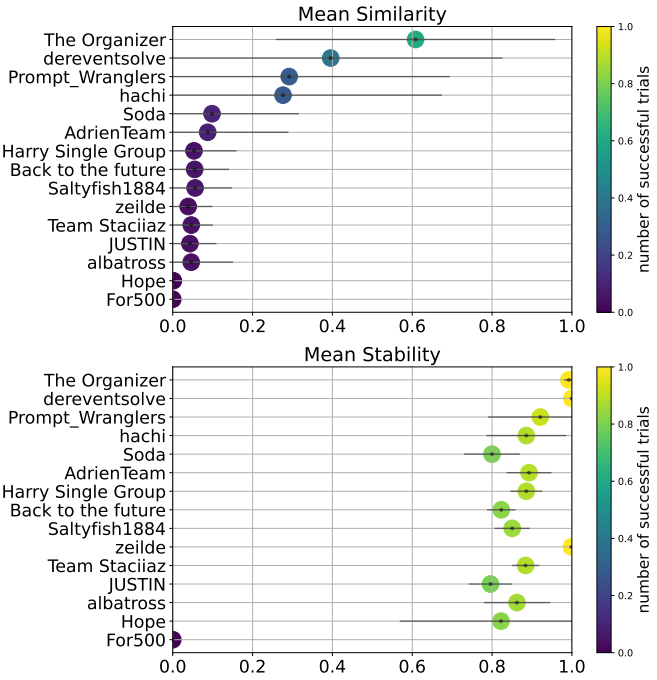


Figure 8: Mean scores for stability and similarity per team.

5 DISCUSSION

Results of the experiments suggest that it is possible to design a replicable competition by carefully selecting the LLM to query and that in the ChatGPT4PCG competition, ChatGPT-3.5 is memorizing from the examples provided in prompts rather than generalizing. Redesigning prompt-based PCG competitions and investigating prompt generalization are both rich areas for future work.

Figure 8 shows the scores calculated for $n = 100$ responses per letter per prompt. It separates the stability and similarity scores for the competition entries with the weightings removed to reduce ambiguity. While most prompts score around 80% or above in stability score shown in the bottom chart, most prompts score between zero and 20% in similarity. In fact there is a sense in which rank differences are nearly entirely decided by similarity scores.

However the top-ranked prompt designed by the organizers of the competition achieves only a 60% average similarity score. One possible cause for the low similarity scores may be due to the training data, which is a collection of handwritten letters. Figure 9 shows block letters clearly representing 'A' and 'B,' yet the softmax probability of these target letters is 0.00 for 'A' and 0.07 for 'B.' These letters would both be classified as H with probability 0.99 and 0.81 percent chance. Similarly, the block letter G is assigned a 0.40 chance of being an 'E', 0.29 chance of 'Z', 0.15 chance of I, and 0.04 chance of F. While the letter 'C' would be a reasonable mistake, neither 'G' nor 'C' are recognized with any probability. The classifier is similarly confused with 'X', guessing 0.37 chance of 'F', 0.33 chance of 'H', 0.09 'Z', 0.03 'I', and 0.02 'K.' Given the chances of misclassification, score is likely largely impacted by the quality of the classifier. Future work will address training a classifier on a more appropriate dataset for the task.

Controllability is an attractive feature of any level generator, and is in particular something the prompt-based PCG paradigm

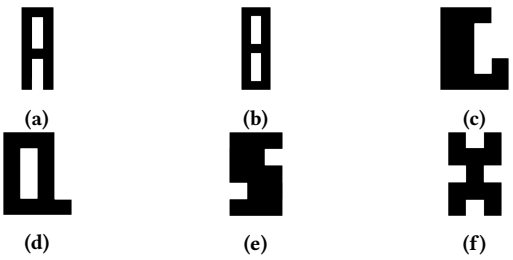


Figure 9: Block Letters: 'A', 'B', 'G', 'Q', 'S', 'X'

should afford. While it may be the case that there are specific prompting techniques to promote controllability [15, 18], state-of-the-art approaches claims that users must be specific about the exact number of game elements that they specify in the prompt rather than communicating with the LLM more qualitatively. In a sense the finding reduces the affordances of prompting to those of parameterized PCG methodologies [4, 8]. A potential downside of such a mapping to parameterized PCG is the impact of the diversity of the level generation because the generator has explicit prompt to include certain amounts of elements. While informal experiments suggest that combining parameterization with more abstract and not readily achievable evaluation criteria (e.g., level elevation [15] or path length [18]) may guide users toward a balance between controllability and diversity, a formal study is one direction for future work.

6 CONCLUSION

This paper takes a deep-dive into the ChatGPT4PCG competition and explores whether the competition is replicable and whether prompts are generalizing or simply giving to ChatGPT-3.5 a memorization cheat-sheet. For the replication experiment, the competition code is run locally and six months after the competition was held. While many prompts earn similar scores, two of the 15 ranked entrants do considerably better in the replication experiment than in the original competition. One reason for the improved performance is the number of valid levels that could be generated and scored. The 'Prompt_Wranglers' prompt was missing important template information in ChatGPT-3.5 responses that led to generating invalid levels and a score of `0.00`. This score was part of what motivated the replication study.

In the generalization experiment, the number of example solutions provided in the prompts is varied by analogy to the concept of few-shot prompting in the LLM literature. The best-performing prompt in the original competition included 26 explicit solutions, which reduces level generation to selecting one of 26 prompt-provided answers rather than generalizing to new, unseen problems. When restricted to the few-shot setting, the prompts score lower than they did in the competition. We conclude that generalization in prompt-based PCG remains an open problem. One takeaway from this study is that future prompt-based PCG experiments should be designed to test generalization as a goal of the competition.

Finally, replicating results is problematic for a competition that queries a public, closed-source LLMs such as ChatGPT: Many things can change, and it is difficult to isolate variables. Our replication experiments, despite reusing the competition scripts, found much better performance with two of the fifteen non-disqualified prompts that were submitted to the first ChatGPT4PCG competition, when compared to the official results. Since one of these prompts was our submission, we were able to analyze it in more detail and attempt to understand reasons for the discrepancy. But we were unable to similarly conclude with any confidence why the other prompt, 'hachi', scored much better in our replication than in the original competition. Our suggestion is that that researchers using LLMs, especially closed API-based LLMs, for prompt-based PCG research may wish to save and document (at least in online supplemental material) more details of the experiments than has been typical, perhaps even erring on the side of pedantically hoarding all inputs and outputs whatsoever. For example, if in addition to the prompts, we had all ChatGPT responses as run at the time of the competition, we would have been more easily able to compare original responses to responses in our attempted replication.

REFERENCES

[1] ChatGPT4PCG. 2023. ChatGPT4PCG Resources. Accessed 2024-04-26. https://chatgpt4pcg.github.io/2023/resources.

[2] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv preprint arXiv:2010.11929* (2020).

[3] Lucas Ferreira and Claudio Fabiano Motta Toledo. 2014. A search-based approach for generating Angry Birds levels. In *2014 IEEE Conference on Computational Intelligence and Games*. https://doi.org/10.1109/CIG.2014.6932912

[4] Britton Horn, Steve Dahlskog, Noor Shaker, Gillian Smith, and Julian Togelius. 2014. A comparative evaluation of procedural level generators in the Mario AI framework. In *Proceedings of the Foundations of Digital Games*. Society for the Advancement of the Science of Digital Games.

[5] Felix Kreuk, Gabriel Synnaeve, Adam Polyak, Uriel Singer, Alexandre Défossez, Jade Copet, Devi Parikh, Yaniv Taigman, and Yossi Adi. 2023. AudioGen: Textually Guided Audio Generation. In *The Eleventh International Conference on Learning Representations, ICLR 2023*. OpenReview.net. https://openreview.net/pdf?id=CYK7RfcOzQ4

[6] Robert Tjarko Lange, Yingtao Tian, and Yujin Tang. 2024. Large Language Models As Evolution Strategies. *arXiv preprint arXiv:2402.18381* (2024).

[7] Elliot Meyerson, Mark J Nelson, Herbie Bradley, Adam Gaier, Arash Moradi, Amy K Hoover, and Joel Lehman. 2023. Language Model Crossover: Variation through Few-Shot Prompting. *arXiv preprint arXiv:2302.12170* (2023).

[8] Arash Moradi Karkaj and Shahriar Lotfi. 2022. Using estimation of distribution algorithm for procedural content generation in video games. *Genetic Programming and Evolvable Machines* 23, 4 (2022), 495–533.

[9] National Academies of Sciences, Engineering, and Medicine. 2019. *Reproducibility and Replicability in Science*. The National Academies Press.

[10] OpenAI. 2023. GPT-4 Technical Report. *CoRR* abs/2303.08774 (2023). https://doi.org/10.48550/ARXIV.2303.08774 arXiv:2303.08774

[11] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical Text-Conditional Image Generation with CLIP Latents. *CoRR* abs/2204.06125 (2022). https://doi.org/10.48550/ARXIV.2204.06125 arXiv:2204.06125

[12] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Bjorn Ommer. 2022. High-Resolution Image Synthesis with Latent Diffusion Models. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 10674–10685.

[13] Téo Sanchez. 2023. Examining the Text-to-Image Community of Practice: Why and How do People Prompt Generative AIs?. In *Proceedings of the Conference on Creativity and Cognition*. 43–61.

[14] Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. 2023. Quantifying Language Models' Sensitivity to Spurious Features in Prompt Design or: How I learned to start worrying about prompt formatting. *arXiv preprint arXiv:2310.11324* (2023).

[15] Shyam Sudhakaran, Miguel González-Duque, Claire Glanois, Matthias Freiberger, Elias Najarro, and Sebastian Risi. 2023. MarioGPT: Open-Ended Text2Level Generation through Large Language Models. In *Proceedings of NeurIPS 2023*.

[16] 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.

[17] Pittawat Taveekitworachai, Febri Abdullah, Mury F Dewantoro, Ruck Thawonmas, Julian Togelius, and Jochen Renz. 2023. ChatGPT4PCG competition: character-like level generation for science birds. In *Proceedings of the 2023 IEEE Conference on Games*.

[18] Graham Todd, Sam Earle, Muhammad Umair Nasir, Michael Cerny Green, and Julian Togelius. 2023. Level Generation Through Large Language Models. *CoRR* abs/2302.05817 (2023). https://doi.org/10.48550/arXiv.2302.05817 arXiv:2302.05817

[19] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems* 35 (2022), 24824–24837.

[20] Tongshuang Wu, Ellen Jiang, Aaron Donsbach, Jeff Gray, Alejandra Molina, Michael Terry, and Carrie J Cai. 2022. Promptchainer: Chaining large language model prompts through visual programming. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*. 1–10.

[21] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2023. Large Language Models as Optimizers. *arXiv preprint arXiv:2309.03409* (2023).