# Large Language Models and Games: A Survey and Roadmap

Roberto Gallotta[1] *Graduate Student Member, IEEE*, Graham Todd[2] *Graduate Student Member, IEEE*, Marvin Zammit[1] *Graduate Student Member, IEEE*, Sam Earle[2] *Graduate Student Member, IEEE*, Antonios Liapis[1] *Member, IEEE* Julian Togelius[2] *Senior Member, IEEE,* and Georgios N. Yannakakis[1], *Fellow, IEEE*
[1]: Institute of Digital Games, University of Malta, Msida, Malta
[2]: Tandon School of Engineering, New York University, New York, USA
roberto.gallotta@um.edu.mt, gdrtodd@nyu.edu, marvin.zammit@um.edu.mt, se2161@nyu.edu, antonios.liapis@um.edu.mt, julian@togelius.com, georgios.yannakakis@um.edu.mt

**Abstract**—Recent years have seen an explosive increase in research on large language models (LLMs), and accompanying public engagement on the topic. While starting as a niche area within natural language processing, LLMs have shown remarkable potential across a broad range of applications and domains, including games. This paper surveys the current state of the art across the various applications of LLMs *in* and *for* games, and identifies the different roles LLMs can take within a game. Importantly, we discuss underexplored areas and promising directions for future uses of LLMs in games and we reconcile the potential and limitations of LLMs within the games domain. As the first comprehensive survey and roadmap at the intersection of LLMs and games, we are hopeful that this paper will serve as the basis for groundbreaking research and innovation in this exciting new field.

**Index Terms**—Large Language Models, Digital Games, Video Games, Survey, Generative Text, Gameplaying, Procedural Content Generation, Generative AI.

## 1 Introduction

Five years ago, autoregressive language modeling was a somewhat niche topic within natural language processing. Training models to simply predict text based on existing text was considered of primarily theoretical interest, although it might have applications as writing support systems. This changed drastically in 2019 when the Generative Pre-trained Transformer 2 (GPT-2) model was released [1]. GPT-2 demonstrated convincingly that transformer models trained on large text corpora could not only generate surprisingly high-quality and coherent text, but also that text generation could be controlled by carefully prompting the model. While not the first autoregressive model [2], [3], GPT-2 was the first of the "large" models, and as such we use it here as cutoff mark (see also Section 2). Subsequent developments, including larger models, instruction fine-tuning, reinforcement learning from human feedback [4], and the combination of these features in ChatGPT in late 2022, turbocharged interest in large language models (LLMs). Capabilities of LLMs were seemingly unbounded, as long as both problem and solution could be formulated as text.

LLMs are currently a very active research field. Researchers are focused on improving the capabilities of LLMs while reducing their compute and memory footprint, but also on understanding and learning to harness the capabilities of existing LLMs. Informed opinions on the ultimate capabilities of LLM technology vary widely, from the enthusiastic [5] to the pessimistic [6], [7]. Our aim is to approach the topic from somewhere in-between these two perspectives: optimistic with respect to the potential of LLMs and realistic with respect to their technical, theoretical, and ethical shortcomings.

Games, including board games and video games, serve both as a source of important benchmarks for research in Artificial Intelligence (AI) and as an important application area for AI techniques [8]. Almost every game utilizes some kind of AI technology, and we are currently in an exploratory phase where both developers and researchers try to figure out how to best make use of recent advances in this field [9].

In this paper, we set out to chart the impact LLMs have had on games and games research, and the impact they are likely to have in the near- to mid-term future. We survey existing work from both academia and (mostly independent) game creators that use LLMs *with* and *for* games. This paper does not set out to capture modern advances in LLM technology or algorithms for training LLMs. Not only do such resources exist [10], but the breakneck speed of technical advances in this field will likely make our writeup obsolete in a year or so. Instead, we focus on work that leverages LLMs in games and propose a range of roles that the LLM can take in the broader ecosystem of games (both within the game and beyond). We lay out promising future directions for efforts to use LLMs in games, and discuss limitations (both technical and ethical) that should be addressed for a brighter future of LLM research in games.

It is important to note that this survey emerges from the top down, based on our expertise in AI and games [8], and extensive work on most topics covered by this paper. The focus of the paper, in Section 3, is built from our own

typology and supported where possible by academic and non-academic work. While a bottom-up approach via e.g. keyword search through general paper repositories is valuable [11], [12], [13], the process would lead to a very different type of paper. Indicatively, this approach was followed by Yang *et al.* [14] who investigated current uses of GPT models in video games by searching articles in ACM, IEEE Xplore, Springer, and AAAI with keywords "game" and "GPT". Instead, we have attempted to conduct a comprehensive manual review of all recent proceedings from the major conferences in AI and games[1], and the IEEE Transactions on Games for work relevant to the themes of this paper.

## 2 A Note on Terminology

This paper concerns the intersection between games (board games, video games, or other), and large language models. But what exactly is an LLM?

Broadly speaking, an LLM is a model that is trained on text in order to be able to reproduce text in response to other text. But this definition is overly broad, as it would include Shannon's original $n$-gram models from 1946 [15], rudimentary recurrent neural networks from the early 1990s [16], and the Tegic T9 text prediction system that would help you write text messages on your Nokia 3210.

What distinguishes LLMs from other text generative models is mainly that they are *large*. But which model size is considered large enough? In 2019 emerging models such as BERT [2] and ERNIE [3] showcased significant advances in language modeling, but LLMs became a well-recognized term with the introduction of OpenAI's GPT-2 [1], whose various versions have between 117 millions and 1.5 billions of parameters. Because of the association between the term LLM and the GPT-class of models, we will use the size of GPT-2 as a soft cutoff on the type of models we consider LLMs; we are concerned with models of few hundred million parameters or more. Each subsequent iteration of the GPT family features an increased number of parameters, and larger and more diverse training corpora.

Another distinct trait of LLMs is their architecture. While language models could in principle be based on various architectures, including Long Short-Term Memory (LSTM) networks [17], the current LLM landscape is dominated by variants of the transformer architecture, a type of neural network introduced in 2017 [18]. This model became very influential because of what was perceived as a quantum leap in output quality compared to previous models. In this survey we rely primarily on LLMs employing this architectural basis.

The last feature of LLMs we consider is their versatility across a wide range of tasks with minimal or no fine-tuning or retraining. This capability represents a significant shift; since the release of GPT-3.5, LLMs have evolved from primarily autoregressive predictive text models to pre-trained, general-purpose conversational models.

It is important to note that LLMs are by no means limited to the GPT family of models. There is by now a large variety of LLMs of varying size and capabilities, including open-source models such as Mistral [19] and the Llama [20] family, which can be fine-tuned, run locally, and even be embedded in games' runtimes.

One could also argue that the definition is somewhat narrow, as many modern LLMs are multimodal models, meaning that they can take as input and/or produce as output modalities other than text. In particular, many modern LLMs can process and produce images. This is often achieved through combining the core transformer network with a visual encoder network for input and a latent diffusion model for output. Examples include GPT-4V [21] and the open-source Llava [22]. In this paper, we consider large multimodal models (LMMs) [21] as long as they retain their ability to both consume and produce text.

This survey will not concern itself with AI and machine learning techniques that are not LLMs as defined above. In particular, we will not be covering the large literature on game playing and content generation using machine learning methods [23] that does not use textual input and output. We will, however, occasionally mention some of that work where relevant, in particular to help provide historical context.

## 3 Roles of LLMs in Games

Past attempts at a typology for AI in games focused on three roles the AI can take in a game: to play a game, to design a game, or to model the (human) players [8]. LLMs are typically presented as conversational agents, which often invites the public to give them anthropomorphic qualities —such as reasoning and creativity. We follow these trends when considering the roles an LLM can be called to play within the game or within the game development process. An LLM can operate within the game as a player (replacing a human player while imitating their goals), as a non-player character such as an enemy or interlocutor, as an assistant providing hints or handling menial tasks for a human player, as a Game Master controlling the flow of the game, or hidden within the games' ruleset (controlling a mechanic of the game). There are however other roles an LLM can play outside of the game's runtime, such as a designer for the game (replacing or assisting a human designer) or as analyst of the gameplay data of the playerbase. Finally, the LLM can interface with a player or an audience in different ways, acting as a commentator of an ongoing play session (during runtime) or a reteller of past game events in some narrative form (outside runtime). Some of these roles (autonomous player, autonomous designer) are prominent in the broader AI and games research [8] and LLM research has targeted them extensively, while some of the other roles have been toyed with in exploratory research. The following sections present the roles themselves, surveying research undertaken for each role, while we identify gaps and opportunities for future research in Section 4.

### 3.1 Player

How can an LLM play a game? Fundamentally, LLM players require some transformation from their typical output space (i.e. sequences of tokens) into the input space of the game. In addition, aspects of the game and its current state must

1. IEEE Conference on Games, Foundations of Digital Games conference, Artificial Intelligence and Interactive Digital Entertainment conference.

be provided to the LLM in some form in order for it to play at a reasonable level. Depending on the game itself, these mappings might be intuitive or complex. We identify three general classes of games to which LLM players are well suited: (a) games where states and actions can be compactly represented as sequences of abstract tokens, (b) games where the main input and output modalities are natural language, and (c) games for which external programs can control player actions via an API.

The first class of games mostly includes turn-based board games (e.g. Chess), since the discrete set of board positions and moves is more easily transformed into a compact representation (e.g. Portable Game Notation [24]) than, for instance, a first-person shooter. By tokenizing sequences of moves taken from a game database, the problem of action selection can then be mapped to the standard autoregressive learning objective on which LLMs are trained —predicting the next move given the context of those that preceded it. Chess [25], [26], [27], Go [28], and Othello [29] have all been used as testbeds for LLM players in this way. This approach allows even more complex game states to be reasoned upon by an LLM player. In Bateni and Whitehead's work [30], for example, the LLM plays the popular video game Slay the Spire (Mega Crit, 2017), understanding synergies between cards based solely on their description and adapting to changes in gameplay rules. However, board games are not the only kind of game that can be represented as token sequences: the generalist GATO [31] agent can play a variety of Atari games at human or near-human levels by processing visual inputs as sequences of pixel values in raster order. Pixel values are interleaved with separator tokens and previous actions, allowing the model to accurately predict the appropriate game action in a dataset of human play traces. It is possible that continued improvement in transformer models that capture both spatial and visual dynamics [32], [33] could allow for a similar approach to scale to even more complex games. However, such approaches require large datasets of gameplay videos that may be comparatively more difficult to collect. In addition, we note that reliance on human gameplay traces as the basis for learning may make it more difficult for an LLM player to reach superhuman performance without leaps in terms of reasoning and generalization (see Section 5).

The second class of games most obviously includes text adventure games such as Zork (Infocom, 1977), where game states are presented as natural language descriptions and the game is already equipped with a parser to handle natural language responses. This means that LLMs can be queried for game actions in a way that still leverages their large-scale pre-training on natural language text. The earliest application of LLMs to these kinds of text games is CALM [34], a GPT-2 system fine-tuned on a dataset of human gameplay transcripts collected from a variety of text adventure games. The model is trained to predict the natural language string provided by human players given the context of previous states, actions and information about the avatar (e.g. their inventory). To actually play a game, the trained language model generates multiple candidate actions and deep reinforcement learning (RL) is used to optimize a policy that selects actions from among the candidates. At the time of its publication, this RL component was necessary because the LLM alone was not capable of generalizing well to unseen games or situations [34]. However, a more recent investigation of ChatGPT as a Zork player has indicated that LLM performance is improving [35]. In a preliminary experiment, Yao et al. [34] show that the performance of ChatGPT can approach that of existing algorithms for text game playing, as long as a human interlocutor remains in the loop to assist the model (e.g. by reminding it of actions it has already tried). However, there is obviously much room for improvement in directly applying LLMs to text games in this way. Additionally, the ability for LLMs to play entirely novel, niche, or unseen text games (especially important given the likelihood that such systems encounter walkthroughs or playtraces of popular text games during their training) remains largely unexplored.

In a similar vein, inductive biases from large language models can be applied to help guide the policies of agents trained with other methods. For instance, the GALAD system [36] uses a pre-trained LLM to guide an agent towards morally acceptable actions in text games from the Jericho suite [37], while the MOTIF system [38] learns an intrinsic reward function for NetHack (NetHack, 1987) by mining preferences between game states from an LLM.

Text adventure games are not the only cases where natural language input and output are used for playing: many board games operate via player negotiation. CICERO [39] leveraged LLMs for playing the deal-making and subterfuge game Diplomacy (Avalon Hill Games, 1976). CICERO builds from a pre-trained LLM and is fine-tuned on a large corpus of Diplomacy transcripts. Throughout the game, samples from the model are sent to other players and the various dialogue transcripts are collected to condition the potential action. CICERO is further trained to condition its outputs on specific game intents (inferred from the transcripts and added as additional context during training). In order to select an action, CICERO uses a "strategic reasoning module" that predicts the actions of other players, using a value and policy function learned from self-play. Diplomacy is an interesting game in part because the action space is split between natural language utterances and a more standard set of moves on a discrete game board, and CICERO demonstrates how an LLM can be integrated as part of a larger system for high-level play.

Finally, we consider games for which a robust API exists. This is less of a kind of game in the sense of its style or mechanics, and more a fact about its popularity or its ease of implementation. An API is an important attribute because it allows LLMs to act as players not by directly generating actions, but by producing programs that act as policies Improvements in the code generation abilities of LLMs have allowed them to write small programs that can produce actions given game states without further intervention from the model. For instance, the VOYAGER system [40] leverages the code generation abilities of GPT-4 to play Minecraft (Mojang Studios, 2011) by interacting with the popular Mineflayer API. Using a sophisticated chain of prompts, VOYAGER generates blocks of code that leverage calls to the API in order to execute high-level "skills" (e.g. "Attack nearest zombie") that are automatically converted into low-level game inputs (e.g. mouse movements and key presses). GPT-4 is also used as a high-level goal generator