Instruction-Driven Game Engine: A Poker Case Study

Hongqiu Wu[1,2,3†] **and Xingyuan Liu**[1,2,3†] **and Yan Wang**[4*] **and Hai Zhao**[1,2,3*]
[1]Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]Key Laboratory of Shanghai Education Commission for Intelligent Interaction and Cognitive Engineering, Shanghai Jiao Tong University
[3]Shanghai Key Laboratory of Trusted Data Circulation and Governance in Web3
[4]Tencent
{wuhongqiu,chloelxy,zhaohai}@sjtu.edu.cn,yanwang.branden@gmail.com

Abstract

The ***Instruction-Driven Game Engine (IDGE)*** project aims to democratize game development by enabling a large language model (LLM) to follow free-form game descriptions and generate game-play processes. The IDGE allows users to create games simply by natural language instructions, which significantly lowers the barrier for game development. We approach the learning process for IDGEs as a *Next State Prediction* task, wherein the model autoregressively predicts the game states given player actions. The computation of game states must be precise; otherwise, slight errors could corrupt the game-play experience. This is challenging because of the gap between stability and diversity. To address this, we train the IDGE in a curriculum manner that progressively increases its exposure to complex scenarios. Our initial progress lies in developing an IDGE for Poker, which not only supports a wide range of poker variants but also allows for highly individualized new poker games through natural language inputs. This work lays the groundwork for future advancements in transforming how games are created and played.

1 Introduction

Game developers dedicate creativity to offer immersive experiences to game players. Players immerse themselves in games and offer valuable feedback to developers. This makes a symbiotic relationship between creators and customers. However, as depicted in the comic from Figure 1, there are disconnections between them, due to diverse preferences of players across age, gender, and cultural backgrounds. Despite the fact that many today's games allow for customization of basic characters and appearances, it is an impossible task for developers to craft every aspect of the game to suit the need of every player. Our study seeks to reconcile such a divide.

Game engines, as the heart of game development, are conventionally driven by programming languages. This technical barrier often deters enthusiasts from realizing their game development dreams. In response, we propose a novel concept: *Instruction-Driven Game Engine (IDGE)*, a game engine enabling anyone to fashion a game through natural language instructions and generating the resultant game-play process. Distinct from recent advancements in video-based games (Bruce et al., 2024; Team et al., 2024b), our focus in this paper is on the text-based game states. We leverage Unity to render these states to visual display.

IDGE is a neural engine, meaning it is built upon neural networks, specifically large language models (LLMs) (Brown et al., 2020; OpenAI, 2023; Touvron et al., 2023; Yang et al., 2023). It is designed to follow a **game script**, a detailed instruction that blueprints the game, e.g. settings, rules, elements, and drive the progression of game-play as interacting with players. IDGEs frame the operation of engines as a *Next State Prediction* task, which autoregressively predicts the next game state based on the user-specified game script, previous game state, and current player action.

Training an IDGE faces the dual challenges of **stability** and **diversity**. The former seeks to provide a stable and precise game-play throughout lengthy contexts, while the latter seeks to follow diverse preferences across the large player base. Unfortunately, we empirically see an ironic twist: the model trained directly from naive game logs is neither stable nor diverse. Therefore, we employ a standard-to-diverse curriculum learning methodology, which gradually introduces complexity into the training process, incrementally enhancing the model's diversity while preserving its stability.

*Corresponding author. † Equal contribution. This research was supported by the Joint Research Project of Yangtze River Delta Science and Technology Innovation Community (No. 2022CSJGG1400), the Joint Funds of the National Natural Science Foundation of China (Grant No. U21B2020).

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing
System Demonstrations, pages 507–519
November 12-16, 2024

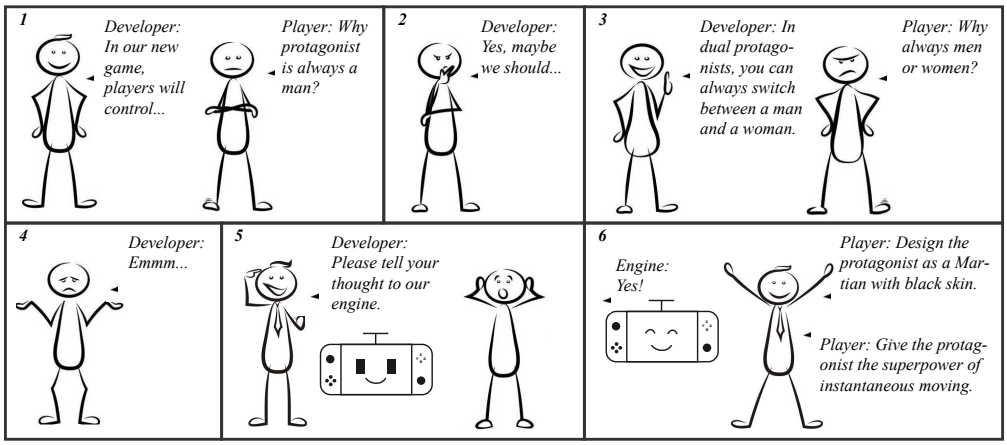

Figure 1: 1: Players were tired of the game's protagonist models. 2, 3: Developers thus created a new mode with dual protagonists. Players still didn't buy it, while they didn't know how to develop games. 4: There were irreconcilable divides between players and developers. 5, 6: Till the advent of the IDGE, it can read the players' mind and let them experience the games immediately.

While it is still on journey from building an IDGE capable of producing AAA games, this paper provides an initial progress on **Poker**, a worldwide card game, e.g. *Texas hold'em*, *Badugi*. We train the IDGE using data sourced from a poker simulator. We show that the IDGE's understanding of nuanced semantics successfully fills voids left by the simulator program, e.g. generating suits and numbers that never occurred in the training process. Furthermore, the IDGE shows immense promise in generalizing to entirely new games, e.g. handling novel card combinations and battle strategies.

We summarize our paper below: • § 2 introduces the concept of the IDGE and its learning problem; • § 3 discusses the IDGE-style data for poker games; • § 4 proposes the enhanced training techniques.

2 Instruction-Driven Game Engine

In this section, we introduce dialogue-style LLMs as the setup for *IDGEs*. We then formulate the learning problem as *Next State Prediction*.

2.1 From Instruction-Driven Dialogue to Instruction-Driven Game Engine

Most LLMs (Brown et al., 2020; OpenAI, 2023; Touvron et al., 2023; Yang et al., 2023) have been fine-tuned on dialogue-style corpora, where it is endowed with the ability to interact with users. The resultant models can follow a system instruction provided by users and lead to a dialogue process in line with it.

Likewise, an IDGE works through interaction, too. Its system instruction specifically refers to a game script that accurately describes the desired game. In game-play, the IDGE interacts with players (users), concurrently processing player inputs, (e.g. moves, targets), to dynamically generate the game states as responses.

In Figure 2, we demonstrate how a poker IDGE facilitates a variant of *Texas Hold'em*: the player first inputs the game script in natural language. Based on this game script, the IDGE simulates the game-play process with the player state by state. The player performs the action, e.g. check, call, raise, and the engine computes and returns the resultant game state. It is a dialogue-like process and will continue till the game concludes.

2.2 Next State Prediction

Causal language models learn the interplay of words through the autoregressive process of next token prediction (Vaswani et al., 2017; Brown et al., 2020). From a game-play perspective, the minimum component is no single token, but rather each **game state**. A game state is a single frame that contains all real-time game information, e.g. characters, items, missions. Essentially, the task of any game engines is exactly to compute the next state according to the prior ones. Therefore, we may formulate the learning of IDGEs as a *Next State Prediction (NSP)* problem.

Given a sequence of game states $\mathbf{s} = \{s_0, s_1, \cdots, s_T\}$, an IDGE with parameters θ seeks to maximize the likelihood:

$$\sum_{t=1}^{T} \log p_\theta(s_t | s_0, s_1, \cdots, s_{t-1}, x_t, z) \qquad (1)$$

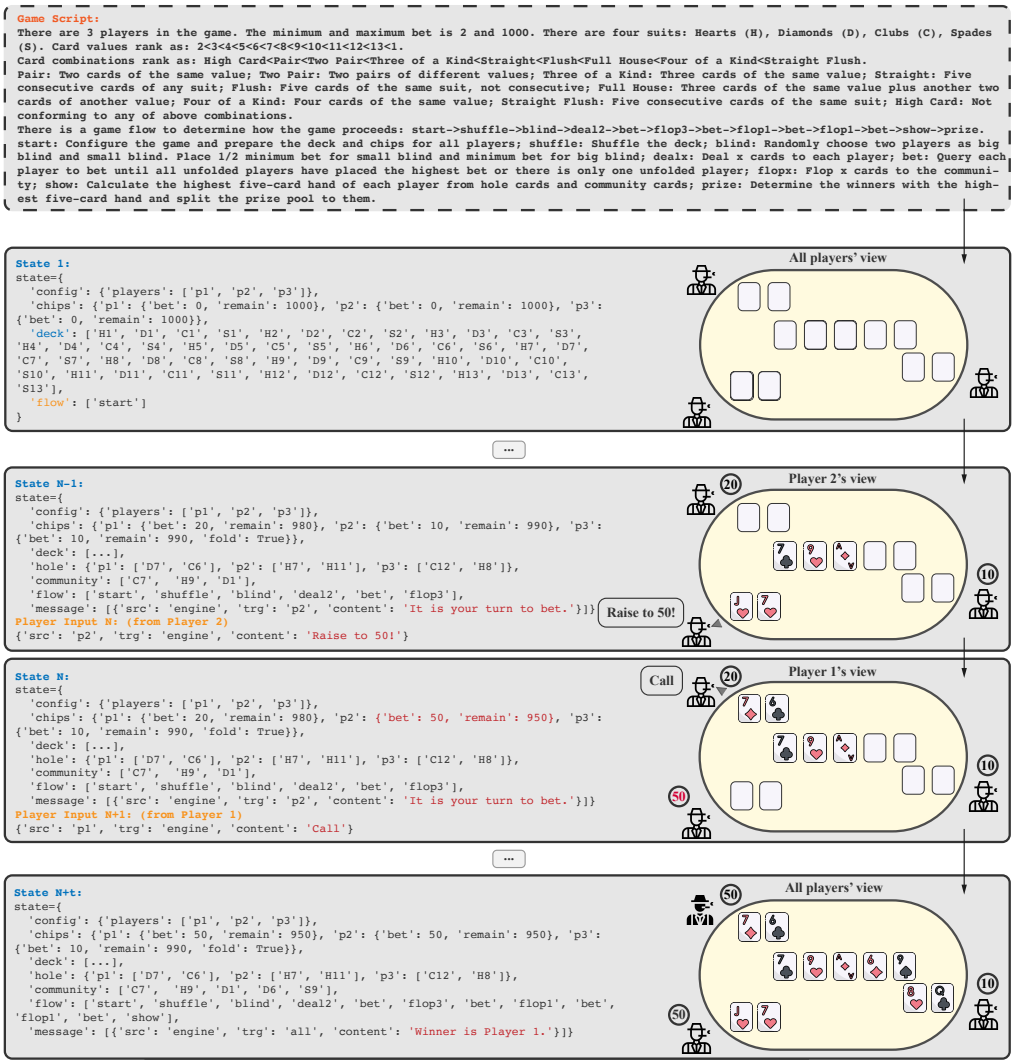

Figure 2: Game-play samples for next state prediction. In the lower half, we illustrate the state prediction circle using NSP. The left side is the input text for the engine from a global view, including all parts that are visible to players as well as those that are not. The right side is the diagram of the game from different players' views.

where x_t refers to the player input at the moment t and z refers to the game script which is global for the entire game. The engine seeks to predict the next state s_t given the prior states $s_0, s_1, \cdots, s_{t-1}$ following z.

A game state is typically far bigger than a token, incurring overflow of inputs and posing challenges for language models in capturing long-range dependencies (Beltagy et al., 2020; Xiao et al., 2024). A more manageable case occurs when it is assumed that each state s_t solely depends on its previous k states. Specifically when $k = 1$, Eq. 1 can be reduced to:

$$\sum_{t=1}^{T} \log p_\theta(s_t | s_{t-1}, x_t, z). \tag{2}$$

While such an independence assumption would incur information loss, a solution is to keep a summary module within the game state.

NSP is a general way to model the process of game-play using a neural engine. However, the practical performance will be limited by models' computational capabilities. For example, it won't be easy for an LLM to handle sophisticated numerical calculation (Wu et al., 2023a), especially for a smaller one. To overcome this weakness, we augment the state prediction process using code modality. The engine is allowed to predict the intermediate code to serve its duty rather than offering the eventual results directly. The prediction will be post-processed by a code interpreter to compute the next state eventually. A toy example is the shuffling of poker cards. It is very hard for a neural model to generate uniformly distributed cards from its inner

representation. To do this, it can define a "shuffle" function and then call it in the next state.

In addition to defining new functions or methods, we allow the engine to call predefined functions, called **core functions**, which are defined in an external core set. These core functions are usually the essential routines that will be frequently used in the game, such as shuffling, ranking of cards in poker. By utilizing core functions, the engine further overcomes the inefficiency of generating repeated content.

The integration of core functions extends IDGEs' functionality and flexibility, enabling them to handle a broader scope of games. This design is akin to the hierarchical architecture in conventional game engines, where the high layers are allowed to call utilities from the core layer.

2.3 Differential State Prediction

The inference complexity of NSP scales quadratically with the sequence length. Therefore, decoding a lengthy game state may fall into trouble. Empirically, the game state only undergoes a slight change between two successive moments t and $t+1$, with the majority of the state remaining the same. This phenomenon can be potentially general across various games when the intervals between states are short. We thus introduce *Differential State Prediction (DSP)*, an efficient variant of NSP, where the engine is simplified to predict solely the difference of two states:

$$\sum_{t=1}^{T} \log p_\theta(\Delta s_t | s_{t-1}, x_t, z) \tag{3}$$

where Δs_t is the difference of s_{t-1} and s_t. DSP is more efficient compared to NSP in most situations, significantly accelerating the inference during game-play. In our experiments, we find that DSP also produces slightly better performance.

To reconstruct s_t from Δs_t and s_{t-1}, there will be a merge function $s_t = M(\Delta s_t, s_{t-1})$. In this work, each game state is implemented as a dict. Hence, M refers to the coding of updating dict elements. The following section will demonstrate concrete examples of NSP/DSP for a poker game.

3 Data for IDGE

Our training data is sourced from two methods. First, we develop a poker simulator and obtain the training data from its game logs. The simulator supports ten representative poker games: *Texas Hold'em*, *Omaha*, *Omaha HL*, *Short-deck Hold'em*, *2-to-7 triple Draw*, *A-to-5 triple Draw*, *2-to-7 single Draw*, *Badugi*, *Badeucey*, and *Badacey*. Additionally, it allows for further configuration of several common poker elements, e.g. type of suits, numbers. By adjusting these elements, one can derive virtually infinite variations beyond aforementioned ten poker games. Moreover, we realize that if the game logs are sampled completely in uniform, the occurrence of some rare states, such as some superior card combinations, would be extremely low. The resultant engine trained on such data may fall short in low-frequency situations, even though the dataset is large. Therefore, we balance the data by up/down-sampling the game logs to ensure that all possible situations occur similarly. After obtaining game logs, we transform each log into a training sample as in Figure 2 for NSP and DSP. Each sample is made up of three parts: the game script z, player input x_t, and game states s_t. If we were to draw an analogy with ChatGPT, they respectively play the roles of the system, user, and assistant.

The second part of the data is generated by GPT3.5. Based on the state prediction data from the simulator, we prompt GPT3.5 to augment the game scripts and generate the corresponding new game states. This process is manually done by skilled prompting, which incorporates scenarios beyond typical poker games, expanding the diversity of training data as a result.

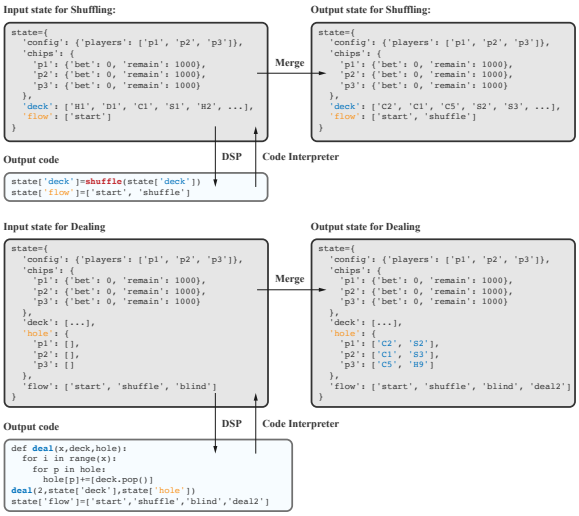

Figure 3: DSP. In the shuffling case, the IDGE calls "shuffle", which is a predefined core function. In the dealing case, it defines a new "deal" function to deal a number of cards to each player one by one. We use a code interpreter to merge the input state and the output code to obtain the next state.

num. of samples	len. of script	len. of input	len. of output NSP → DSP	avg. states
10k	439.8	401.1	404.9 → 135.9	35.3

Table 1: Statistics of training data.

♠ **Game Script** To describe the poker game in natural language, we design a prototype game script. The top part of Figure 2 illustrates the game script for a *Texas Hold'em* variant. We can see that it defines a series of game elements: the number of players in the game, minimum and maximum bet limits, suits and values of single cards, battle strategies, and the game flow. These elements correspond to the configuration of the poker simulator. Particularly, the game flow refers to the procedures this game will go through in order, e.g. bet, deal.

♠ **Game State and Player Input** For the game state and player input, we adopt a dict format as shown on the left side of Figure 2. For instance, "deck" is followed by the remaining cards in the deck, "hole" and "community" is followed by the hole cards of players and the public cards, while "message" is followed by the message sent from the source a to target b. On the right side of Figure 2, we show the diagram of the poker game associated to the left-side game state. In player input N, player 2 chooses to raise the bet. Given state $N-1$, the engine outputs state N, where the chips of player 2 are updated and player 1 is informed to bet since player 3 has folded.

To ensure the independence assumption that each state s_t solely depends on state s_{t-1}, we incorporate the game flow as a summary module in the game state. It specifically caches all past game procedures in order. The engine can thus be navigated to step into the next procedure correctly, regardless of the amount of game-play history.

Data Statistics Table 1 shows the statistics of the training data that we construct, which comprises 10k state prediction samples. Specifically, the average number of states of one game is 35.3, i.e. the number of states for the engine to predict. The output tokens of DSP is much less than that of NSP.

4 Curriculum Learning

Straightforwardly, we could utilize the data generated in § 3 to fine-tune a base model by maximizing Eq. 2/3 and obtain the IDGE. However, the resultant IDGE may struggle with stability and diversity: neither can it accurately predict the next game state nor comprehend the user-specified game script in natural language. Therefore, we devise a progressive curriculum learning process (Bengio et al., 2009), to incrementally enhance the IDGE's diversity while preserving stability.

Warmup: Training on Core Set In § 2, we utilize a set of core functions to facilitate the process of state prediction. Though the model can be exposed to all core functions via fine-tuning, we observe that it struggles to call the core functions properly. This phenomenon is much more severe in unseen contexts. We attribute this to the cold start problem that the model merely memorizes the names of the core functions during training, without knowing their underlying implementation. To this end, we introduce a pre-learning phase to warmup the model. We develop an instruction tuning dataset of 1k samples derived from the core set, where each core function is translated to a natural language instruction and the model is trained to implement the function in a way of instruction following. This phase offers a profound comprehension of the model's usage of core functions.

Standard: Training on Standard Game Scripts The next step is to train the model on the standard data introduced in § 3 by optimizing NSP/DSP. In this phase, the model is forged into an engine, predicting game-play state by state following the game scripts, and is combined with pre-learned core functions organically.

Diverse: Training on Rephrased Game Scripts While the standard data already includes the prototype game scripts, mastering the prototype descriptions can be too restrictive for users. Rather, it is more natural for them to describe their desired games in free-form natural language. Rather than exhaustively crafting new natural language data, we introduce *Segment Rephrasing (SR)*, a technique that rephrases a portion of the game script to encourage the model to follow diverse natural language. Specifically, given a game script, we segment it into chunks and randomly rephrase several of them. To largely keep the semantics intact, there is only a very low probability that the entire script will be rephrased. The rephrasing process is done by GPT3.5. These rephrased game scripts enable the model fully "to customers". In addition, these scripts will be more challenging to understand, which potentially generalizes the model to unseen scenarios. Readers may refer to Table 5 in Appendix B for real human-written examples.

We summarize the training pipeline for the IDGE: 1) train on the core set $\mathcal{D}_{cs}$ (1k); 2) train

by optimizing NSP/DSP on the standard dataset $\mathcal{D}$ (10k); 3) rephrase the standard data $\mathcal{D}_{sr}$ and train on the sum of $\mathcal{D}$ and $\mathcal{D}_{sr}$ (20k).

The **warmup**, **standard**, and **diverse** process correspond to the easy, medium, and hard curriculum. It serves for a smooth transfer of the IDGE from standardization to diversity.

5 Experimental Results

In this section, we evaluate the IDGE in two scenarios. The former is automatically generated by our simulator, which can be considered as a test set that has the same distribution as the training set. The latter resembles the real-world situations, where proficient poker players are directly enlisted as annotators to create new game scripts. Subsequently, the test data is obtained by playing the games online by themselves with the IDGE.

5.1 Training and Evaluation Setup

We develop the IDGE based on CodeGemma-7b (Team et al., 2024a)[1]. CodeGemma is a code LLM that is additionally pre-trained on large code corpora. We find that CodeGemma works better than similar-sized natural language models like LLaMA3 (Dubey et al., 2024). We train each model using LoRA (Hu et al., 2022) with $r = 8$, $\alpha = 32$, and the optimal learning rate in 1.5e-4 and 3e-4. The warmup of the learning rate is set to 30 steps and the total batch size is set to 8 on 8 chips. For each curriculum, we train 3 epochs. To ensure the stability of outputs, we leverage greedy decoding.

• **In-domain evaluation**: The model has been exposed to a broad range of variants based on ten existing poker games during training. We sampled some unseen variants of these ten games from the poker simulator for evaluation. Then, we program some random players that randomly select an action as their input to interact with the IDGE. This manner allows for a quick and automatic assessment of the IDGE's basic performance as well as the effectiveness of training methods. Specifically, each type of games is played for 20 rounds. There are totally 200 rounds of games in the in-domain test set. The state prediction accuracy is determined through two steps. First, we compare the predicted code snippet and the ground truth. If not exactly matched, we execute both snippets on the input state respectively and then compare two outputs.

[1] https://huggingface.co/google/codegemma-7b-it

• **Out-of-domain evaluation**: The in-domain evaluation is limited to a number of predefined poker games with configurable essential elements. To evaluate our the IDGE's performance in scenarios more closely aligned with the real world, we further recruit 5 proficient poker players as our engine testers. Each of them is asked to create 1∼2 new poker games based on their personal preferences and craft the game script using natural language. They are free to tailor the game scripts, for example, crafting the entirely new elements and strategies not found in existing poker games. Subsequently, we invite them to play 10 rounds of the game with distinct configurations for each new game by themselves and record all player inputs and game states throughout the game-play. This forms our out-of-domain test set that comprises 8 distinct game scripts and 80 rounds of games.

5.2 In-Domain

Round-level Table 2 shows the round-level success rates of a number of fine-tuned models. The success rate is counted if the engine correctly handles all states in a round. The results of CodeGemma from NSP to DSP suggest the advantage of predicting the difference of two states, which results in both accuracy and efficiency boost. The best results occur when the model undergoes segment rephrasing (SR) and the full curriculum (CS + SR) respectively. The resultant CodeGemma achieves 100% success rates on all ten poker variants. This suggests the effectiveness of SR to enhance the model's understanding on the game scripts. In the following, we will show that SR is more important in the face of out-of-domain games.

State-level We also introduce GPT4 as a strong baseline in our experiment, which is prompted with additionally five in-context samples (5-shot). Surprisingly, in 200 rounds of games, it is unable to successfully complete any single round. One might question why GPT4 completely fails in this task, significantly behind fine-tuned CodeGemma-7b. To conduct a more in-depth analysis, we compute the state-level accuracy in Table 3. We find that, though GPT4 is strong in programming, it performs badly in managing nuanced poker cards. For example, it is very likely to mess up the order, hallucinating new cards or missing some of them. This drawback is pronounced in `deal` and `show`. In contrast, `deal` is a much easier task for humans. We conjecture that current LLMs have not been exposed to highly sophisticated data and tasks

	Texas	*Omaha*	*Om. HL*	*Short.*	*27 triple*	*A5 triple*	*27 single*	*Badugi*	*Badeucey*	*Badacey*
NSP	✓	✓	18/20	✓	18/20	✓	✓	✓	17/20	18/20
DSP	✓	✓	✓	✓	✓	✓	✓	✓	18/20	18/20
DSP (CS)	✓	✓	✓	✓	✓	19/20	✓	✓	✓	✓
DSP (SR)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
DSP (CS+SR)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 2: Round-level success rates on 10 existing poker variants for 20 rounds. We use ✓ to indicate the 100% success rate. CS and SR refer to the core set and segment rephrasing technique.

	`start` (A)	`blind` (C)	`shuf.` (D)	`deal` (C)	`flop` (C)	`switch` (B)	`bet` (B)	`show` (A)	`prize` (A)
GPT4 (5-shot)	88.0	84.0	✓	31.3	77.6	20.6	78.7	0.0	83.0
CoGem. (5k)	94.0	✓	✓	✓	✓	✓	93.0	88.0	✓
CoGem. (10k)	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 3: State-level performances with different values of training data based CodeGemma-7b (DSP+CS+SR). We use ✓ to indicate the 100% accuracy. We label the difficulty of each type of states from A to D from hard to easy.

	IDGE	IDGE w. SR
MAGIC DEALER	✗	✓
3-CARD DRAW	8/10	✓
6-CARD DRAW	✗	✓
DRAGONIE	1/10	9/10
THREE KINGDOMS	✗	✓
STARDUST	✗	8/10
ODD LOVER	3/10	✓
JOKER HOLD'EM	✗	✓

Table 4: Success rates on out-of-domain games.

as for the IDGEs during their training. The accumulation of errors in all these aspects eventually leads non-fine-tuned models to zero success rates in round-level evaluation. It is important to note that an IDGE should be all-round at each aspect; otherwise, the overall performance will degenerate in a way of **Buckets effect**.

In contrast, for fine-tuned CodeGemma, Table 3 shows that it has performed close to 100% accuracy in most states with only a half of training samples (5k). Such high accuracy correlates positively to its stable round-level performance in Table 2. We notice that CS is particularly beneficial for `show`, where the model is responsible to calculate the hand combinations and compare their strength, the most challenging task in poker games. There are a large number of relevant core functions in this process. Hence, it becomes critical for the model to adapt to core functions in advance.

5.3 Out-of-Domain

Table 5 in Appendix B illustrates the eight scripts created by human players. Most of them are creative new games with a large gap from standard poker. For example, in script 6, the creator defines a group of novel combinations "Stardust X".

Table 4 reports the round-level success rates of our IDGE, fine-tuned based on CodeGemma-7b with and without SR. We first find that the model not underwent SR fails to be fully instructable by players. For example, it cannot understand the tricky dealing process in *Magic Dealer* described in free natural language, though it is a simple variant from standard dealing. In contrast, the model underwent SR treats this with ease. The rephrased samples encourage the model to learn the alignment between prototype game scripts and diverse natural language, thereby better balancing stability and diversity. Additionally, the full IDGE demonstrates remarkable generalizability in the face of novel and unseen games. For example, in *6-card Draw*, the IDGE effectively generalizes from managing 5-card hands to 6-card hands, while in *Dragonie*, which is an upgrade version of *Badugi*, the IDGE learns to pick out cards with distinct suits while determining the consecutiveness of their values. For more challenging *Stardust*, where the creator introduces a series of entirely new cards and combinations, the IDGE successfully passes eight of the ten rounds of the game.

6 Conclusion

This paper introduces the Instruction-Driven Game Engine (IDGE), offering game enthusiasts a brand new game development and game-play experience. The IDGE understands the player-specified game rules and simulates the entire game-play process. We formulate the learning of IDGEs as Next State Prediction and leverage a curriculum learning approach to enhance stability and diversity. Experiments demonstrate our poker IDGE can accurately complete the majority of user-defined games.

Broader Impact

This paper presents the initial progress of IDGE in the case of Poker. Such a paradigm theoretically applies to all types of games. However, our progress is constrained by several bottlenecks.

Inference Latency We have demonstrated that IDGEs go well with turn-based strategy (TBS) games. For real-time strategy (RTS) games, players may make more than one action per second. The inference latency of current LLMs cannot meet the real-time requirements of such games.

Context Window Generally, as games become more complicated, the length of game states increases, posing a challenge to satisfy our independence assumption. This may significantly challenge both the comprehension ability of LLMs and the cache of KV states.

Accessibility The kernel data of most commercial games is not publicly available, which is why we developed a poker simulator to generate the training data for this paper.

We are delighted to observe that there have been continuous advancements in inference frameworks such as vLLM (Kwon et al., 2023), as well as efficient long-text generation methods like StreamingLLM (Xiao et al., 2024) and TempLoRA (Wang et al., 2024). We believe that the ongoing development of LLM technologies will ultimately address the limitations of latency and the context window. Regarding the issue of accessibility, we look forward to more companies providing open interfaces as SC2LE (Vinyals et al., 2017), HOK Arena (Wei et al., 2022) to offer kernel data.

The recent released Delta-Engine (Wu et al., 2024a) is largely inspired from our work. It exclusively focuses on game development. The development process can be ideally eternal, by expanding the engine incrementally. Unlike the IDGE, the delta-engine does not simulate the game-play process. The resultant game-play is rendered by external modules.

References

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, volume 382 of *ACM International Conference Proceeding Series*, pages 41–48. ACM.

Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. 2017. Heads-up limit hold'em poker is solved. *Commun. ACM*, 60(11):81–88.

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. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Jake Bruce, Michael D. Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Bechtle, Feryal M. P. Behbahani, Stephanie C. Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott E. Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando de Freitas, Satinder Singh, and Tim Rocktäschel. 2024. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurélien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Rozière, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Grégoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel M. Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia,

Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, and et al. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Mirjam Palosaari Eladhari. 2018. Re-tellings: The fourth layer of narrative as an instrument for critique. In *Interactive Storytelling - 11th International Conference on Interactive Digital Storytelling, ICIDS 2018, Dublin, Ireland, December 5-8, 2018, Proceedings*, volume 11318 of *Lecture Notes in Computer Science*, pages 65–78. Springer.

Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. 2022. Minedojo: Building open-ended embodied agents with internet-scale knowledge. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Roberto Gallotta, Graham Todd, Marvin Zammit, Sam Earle, Antonios Liapis, Julian Togelius, and Georgios N. Yannakakis. 2024. Large language models and games: A survey and roadmap. *CoRR*, abs/2402.18659.

Akshat Gupta. 2023. Are chatgpt and GPT-4 good poker players? - A pre-flop analysis. *CoRR*, abs/2308.12466.

Senyu Han, Lu Chen, Li-Min Lin, Zhengshan Xu, and Kai Yu. 2024. IBSEN: director-actor agent collaboration for controllable and interactive drama script generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 1607–1619. Association for Computational Linguistics.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *CoRR*, abs/2310.06825.

Juho Kim. 2023. Pokerkit: A comprehensive python library for fine-grained multi-variant poker game simulations. *CoRR*, abs/2308.07327.

Heinrich Küttler, Nantas Nardelli, Alexander H. Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. 2020. The nethack learning environment. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. *Preprint*, arXiv:2309.06180.

Ryan Lowe, Abhinav Gupta, Jakob N. Foerster, Douwe Kiela, and Joelle Pineau. 2020. On the interaction between supervision and self-play in emergent communication. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. 2013. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602.

Matej Moravcík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael H. Bowling. 2017. Deepstack: Expert-level artificial intelligence in no-limit poker. *CoRR*, abs/1701.01724.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, Bokai Xu, Zhen Zhang, Yining Ye, Bowen Li, Ziwei Tang, Jing Yi, Yuzhang Zhu, Zhenning Dai, Lan Yan, Xin Cong, Yaxi Lu, Weilin Zhao, Yuxiang Huang, Junxi Yan, Xu Han, Xian Sun, Dahai Li, Jason Phang, Cheng Yang, Tongshuang Wu, Heng Ji, Zhiyuan Liu, and Maosong Sun. 2023. Tool learning with foundation models. *CoRR*, abs/2304.08354.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Noah Ranella and Markus Eger. 2023. Towards automated video game commentary using generative AI.

In *Proceedings of the Experimental Artificial Intelligence in Games Workshop co-located with the 19th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE 2023), Salt Lake City, Utah, USA, October 8, 2023*, volume 3626 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Murray Shanahan, Kyle McDonell, and Laria Reynolds. 2023. Role play with large language models. *Nat.*, 623(7987):493–498.

Weihao Tan, Ziluo Ding, Wentao Zhang, Boyu Li, Bohan Zhou, Junpeng Yue, Haochong Xia, Jiechuan Jiang, Longtao Zheng, Xinrun Xu, Yifei Bi, Pengjie Gu, Xinrun Wang, Börje F. Karlsson, Bo An, and Zongqing Lu. 2024. Towards general computer control: A multimodal agent for red dead redemption II as a case study. *CoRR*, abs/2403.03186.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024a. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

SIMA Team, Maria Abi Raad, Arun Ahuja, Catarina Barros, Frederic Besse, Andrew Bolt, Adrian Bolton, Bethanie Brownfield, Gavin Buttimore, Max Cant, Sarah Chakera, Stephanie C. Y. Chan, Jeff Clune, Adrian Collister, Vikki Copeman, Alex Cullum, Ishita Dasgupta, Dario de Cesare, Julia Di Trapani, Yani Donchev, Emma Dunleavy, Martin Engelcke, Ryan Faulkner, Frankie Garcia, Charles Gbadamosi, Zhitao Gong, Lucy Gonzalez, Kshitij Gupta, Karol Gregor, Arne Olav Hallingstad, Tim Harley, Sam Haves, Felix Hill, Ed Hirst, Drew A. Hudson, Jony Hudson, Steph Hughes-Fitt, Danilo J. Rezende, Mimi Jasarevic, Laura Kampis, Nan Rosemary Ke, Thomas Keck, Junkyung Kim, Oscar Knagg, Kavya Kopparapu, Andrew K. Lampinen, Shane Legg, Alexander Lerchner, Marjorie Limont, Yulan Liu, Maria Loks-Thompson, Joseph Marino, Kathryn Martin Cussons, Loic Matthey, Siobhan Mcloughlin, Piermaria Mendolicchio, Hamza Merzic, Anna Mitenkova, Alexandre Moufarek, Valéria Oliveira, Yanko Gitahy Oliveira, Hannah Openshaw, Renke Pan, Aneesh Pappu, Alex Platonov, Ollie Purkiss, David P. Reichert, John Reid, Pierre Harvey Richemond, Tyson Roberts, Giles Ruscoe, Jaume Sanchez Elias, Tasha Sandars, Daniel P. Sawyer, Tim Scholtes, Guy Simmons, Daniel Slater, Hubert Soyer, Heiko Strathmann, Peter Stys, Allison C. Tam, Denis Teplyashin, Tayfun Terzi, Davide Vercelli, Bojan Vujatovic, Marcus Wainwright, Jane X. Wang, Zhengdong Wang, Daan Wierstra, Duncan Williams, Nathaniel Wong, Sarah York, and Nick Young. 2024b. Scaling instructable agents across many simulated worlds. *CoRR*, abs/2404.10179.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Muhtar Çagkan Uludagli and Kaya Oguz. 2023. Non-player character decision-making in computer games. *Artif. Intell. Rev.*, 56(12):14159–14191.

Nidhi Vakil and Hadi Amiri. 2023. Complexity-guided curriculum learning for text graphs. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 2610–2626. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander Sasha Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom Le Paine, Çaglar Gülçehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy P. Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. 2019. Grandmaster level in starcraft II using multi-agent reinforcement learning. *Nat.*, 575(7782):350–354.

Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John P. Agapiou, Julian Schrittwieser, John Quan, Stephen Gaffney, Stig Petersen, Karen Simonyan, Tom Schaul, Hado van Hasselt, David Silver, Timothy P. Lillicrap, Kevin Calderone, Paul Keet, Anthony Brunasso, David Lawrence, Anders Ekermo, Jacob Repp, and Rodney Tsing. 2017. Starcraft II: A

new challenge for reinforcement learning. *CoRR*, abs/1708.04782.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023. Voyager: An open-ended embodied agent with large language models. *CoRR*, abs/2305.16291.

Yan Wang, D. Ma, and Deng Cai. 2024. With greater text comes greater necessity: Inference-time training helps long text generation. *CoRR*, abs/2401.11504.

Hua Wei, Jingxiao Chen, Xiyang Ji, Hongyang Qin, Minwen Deng, Siqin Li, Liang Wang, Weinan Zhang, Yong Yu, Liu Lin, Lanxiao Huang, Deheng Ye, Qiang Fu, and Wei Yang. 2022. Honor of kings arena: an environment for generalization in competitive reinforcement learning. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Hongqiu Wu, Linfeng Liu, Hai Zhao, and Min Zhang. 2023a. Empower nested boolean logic via self-supervised curriculum learning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 13731–13742. Association for Computational Linguistics.

Hongqiu Wu, Yongxiang Liu, Hanwen Shi, Hai Zhao, and Min Zhang. 2023b. Toward adversarial training on contextualized language representation. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Hongqiu Wu, Zekai Xu, Tianyang Xu, Shize Wei, Yan Wang, Jiale Hong, Weiqi Wu, Hai Zhao, Min Zhang, and Zhezhi He. 2024a. Evolving virtual world with delta-engine. *CoRR*, abs/2408.05842.

Weiqi Wu, Hongqiu Wu, Lai Jiang, Xingyuan Liu, Hai Zhao, and Min Zhang. 2024b. From role-play to drama-interaction: An LLM solution. In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 3271–3290. Association for Computational Linguistics.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Yuzhuang Xu, Shuo Wang, Peng Li, Fuwen Luo, Xiaolong Wang, Weidong Liu, and Yang Liu. 2023. Exploring large language models for communication games: An empirical study on werewolf. *CoRR*, abs/2309.04658.

Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, Fan Yang, Fei Deng, Feng Wang, Feng Liu, Guangwei Ai, Guosheng Dong, Haizhou Zhao, Hang Xu, Haoze Sun, Hongda Zhang, Hui Liu, Jiaming Ji, Jian Xie, Juntao Dai, Kun Fang, Lei Su, Liang Song, Lifeng Liu, Liyun Ru, Luyao Ma, Mang Wang, Mickel Liu, MingAn Lin, Nuolan Nie, Peidong Guo, Ruiyang Sun, Tao Zhang, Tianpeng Li, Tianyu Li, Wei Cheng, Weipeng Chen, Xiangrong Zeng, Xiaochuan Wang, Xiaoxi Chen, Xin Men, Xin Yu, Xuehai Pan, Yanjun Shen, Yiding Wang, Yiyu Li, Youxin Jiang, Yuchen Gao, Yupeng Zhang, Zenan Zhou, and Zhiying Wu. 2023. Baichuan 2: Open large-scale language models. *CoRR*, abs/2309.10305.

Enmin Zhao, Renye Yan, Jinqiu Li, Kai Li, and Junliang Xing. 2022. Alphaholdem: High-performance artificial intelligence for heads-up no-limit poker via end-to-end reinforcement learning. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 4689–4697. AAAI Press.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2020. Freelb: Enhanced adversarial training for natural language understanding. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

A Related Work

A game engine is a fundamental software designed for game development. Famous game engines include Unreal, Unity, CoCos, etc. Pygame is also a simple game engine. We spotlight two crucial properties of a game engine. The first is functionality, i.e. providing a wide variety of basic tools to facilitate the development process. The next is secondary development, i.e. rich and flexible interfaces to allow developers to customize games. In this work, we introduce a new concept, instruction-driven game engine (IDGE), a neural game engine learned on basis of large language models (OpenAI, 2023; Touvron et al., 2023; Jiang et al., 2023; Yang et al., 2023; Qin et al., 2023). As opposed to typical game engines, the IDGE acquires its functionality power by instruction tuning on the core set (Raffel et al., 2020; Ouyang et al., 2022) and allows for low-barrier game development by issuing natural language descriptions.

Some research efforts have explored the AI applications in games (Gallotta et al., 2024), e.g. non-play characters (Shanahan et al., 2023; Uludagli and Oguz, 2023), interactive drama (Wu et al., 2024b; Han et al., 2024), game commentators (Eladhari, 2018; Ranella and Eger, 2023). A great amount of work focuses on AI as players, e.g. for Atari (Mnih et al., 2013), Minecraft (Fan et al., 2022; Wang et al., 2023), StarCraft (Vinyals et al., 2019), NetHack (Küttler et al., 2020; Lowe et al., 2020), Werewolf (Xu et al., 2023); However, our work diverges from all of them in that we treat AI as the playground, attempting to build a game engine that is defined by instructions (game scripts) and game states. The former focuses on the way AI behaves, while the latter focuses on the way AI would react in the face of any possible behaviors from human beings and agents. More recent work comes up with learning for a foundation agent, a single agent with generalizable skills to behave in various environments, e.g. SIMA (Team et al., 2024b), an instruction-driven agent proficient in multiple simulated environments; CRADLE (Tan et al., 2024), a powerful agent capable of playing complex AAA games like Red Dead Redemption 2 by controlling the keyboard and mouse. However, our work targets the IDGE for a specific group of games, Poker, as an initial step for building a foundation IDGE. Poker is a widely studied information game of immense popularity (Bowling et al., 2017; Moravcík et al., 2017; Gupta, 2023; Kim, 2023; Zhao et al., 2022).

In this paper, the entire training cycle for IDGE is a way of curriculum learning (Bengio et al., 2009). Recent studies show the potential of curriculum learning in empowering the language models to tackle more challenging tasks (Vakil and Amiri, 2023; Wu et al., 2023a). The proposed segment rephrasing technique is related to perturbation training (Zhu et al., 2020; Wu et al., 2023b), which smooths the structured natural language in the semantic space.

B Out-of-Domain Game Scripts

C System Demonstration

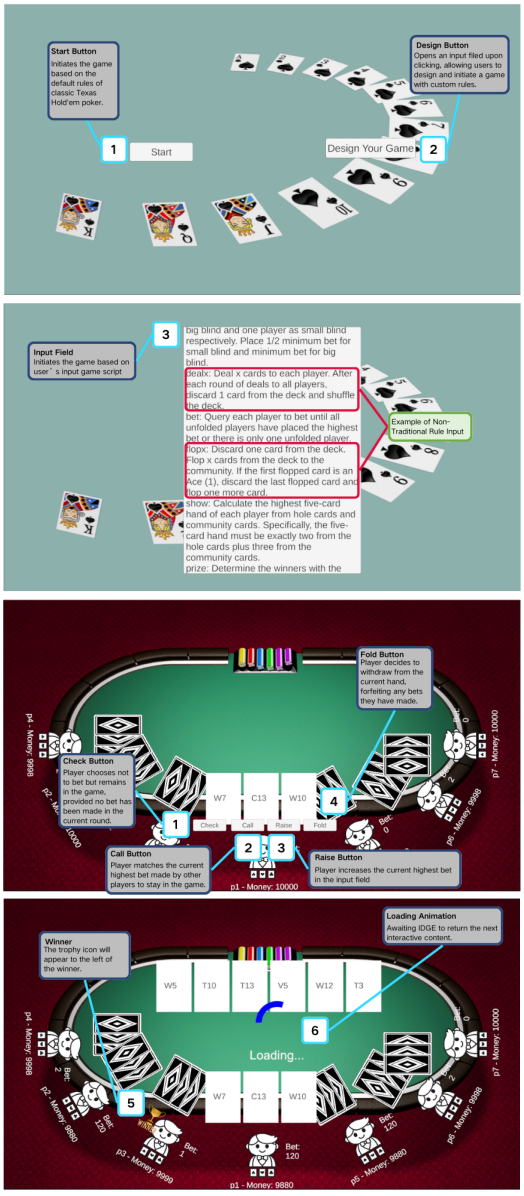

Figure 4: System demonstration of our poker IDGE, developed based on Unity.

Script 1: MAGIC DEALER
The game proceeds in the following order: start the game, shuffling, set blinds, deal 2 cards, bet, reveal 3 cards (the flop), bet,reveal 1 card (the turn), deal 1 card (new deal), bet, show, and finally the prize is distributed. In each dealing phrase, deal x+1 cards to each player. Then randomly discard 1 card from each player's hand and shuffle it back into the deck. In each flop, flop x cards from the deck to the community. Except when x=1, if the first flopped card matches the suit of the last flopped card, flop 1 more.
Script 2: 3-CARD DRAW
Introduce a new game, named "3-card draw". In this game, there are 3 suits, H, D, C, and each player is dealt with a 3-card hand.There are 6 possible combinations of hand. Pair: Two cards of the same value; Three of a Kind: Three cards of the same value; Straight: Three consecutive cards of any suit; Flush: Three cards of the same suit, not consecutive; Straight Flush: Three consecutive cards of the same suit; High Card: Not conforming to any of above combinations.
Script 3: 6-CARD DRAW
Introduce a new game "6-card draw". In this game, there are four suits, Hearts (H), Diamonds (D), Clubs (C), Spades (S). In addition, define two new combinations with 6 cards in hand. Three Pair: there are three pairs of distinct numbers, e.g. D8, H8, C10, H10, H12, D12. Big House: there are two pairs of three of one kind, e.g. H8, C8, S8, C12, H12, D12. All combinations rank as: High Card<Pair<Three of a Kind<Straight<Flush<Full House<Three Pair<Big House<Straight Flush.
Script 4: DRAGONIE
There are four original suits: Hearts (H), Diamonds (D), Clubs (C), Spades (S). There is an additional superior suit: Loong (L). The suits rank as: L>H=D=C=S. Card values rank as: 1<2<3<4<5<6<7<8<9<10<11<12<13. Introduce a new ranking strategy: "Dragonie". For each player with four hole cards, pick out the consecutive cards of distinct suits. Dragonie refers to the four-card hand where four cards are of consecutive cards as well as of distinct suits. In this case, the valid cards are four. In the case that there are three consecutive cards of distinct suits, the valid cards are three. Dragonie>three valid cards>two valid cards>one valid cards. To compare the same number of valid cards, the lowest one is the best.
Script 5: THREE KINGDOMS
These new poker game is called "Three Kingdoms". There are three distinct suits: Shu Han (S), Cao Wei (W), and Dong Wu (D). Each player will be dealt with four hole cards. The biggest hand is the one where at least one of all three kingdoms (suits) is present, call it "Three Kingdoms". The second biggest is the one where at least two kingdoms is present, "Two Kingdoms". The rest of the situations belong to Hard Card. In these game, highest cards are preferred when comparing two hands of the same combination.
Script 6: STARDUST
There are ten special cards: "Stardust" in the deck (represented as *). These cards are of none suit and none value. In hand with one Stardust card, the required number of cards to form a straight or flush will be one less, and is greater than a normal straight or flush. The hand with more than one Stardust, will be reduced to High Card. In detail, Stardust Straight: Four consecutive cards of any suit, plus a Stardust (*); Stardust Flush: Four cards of the same suit, not consecutive, plus a Stardust (*); Stardust Straight Flush: Four consecutive cards of the same suit, plus a Stardust (*). High Card<Pair<Three of a Kind<Straight<Stardust Straight<Flush<Stardust Flush<Straight Flush<Stardust Straight Flush.
Script 7: ODD LOVER
In this game, odd values (1, 3, 5, 7, 9) are greater than even values (2, 4, 6, 8, 10). They rank as: 2<4<6<8<10<1<3<5<7<9. Card combinations rank as: High Card<Odd Straight<Odd Flush<Odd Straight Flush. Odd Straight: Five consecutive odd values of any suit, e.g. 1, 3, 5, 7, 9; Odd Flush: Five odd values of the same suit, not consecutive; Odd Straight Flush: Five consecutive odd values of the same suit; High Card: Not conforming to any of above combinations.
Script 8: JOKER HOLD'EM
There are four suits: Hearts (H), Diamonds (D), Clubs (C), Spades (S). Card values rank as: 2<3<4<5<6<7<8<9<10<J<Q<K<1. In addition, there are two special Joker cards represented as J1 and J2, which can be treated as any suit and value. Three of a Kind: Three cards of the same value. Straight: Five consecutive cards of any suit. Flush: Five cards of the same suit, not consecutive. Full House: Three cards of the same value plus another two cards of another value. Four of a Kind: Four cards of the same value. Five of a Kind: Five cards of the same value (possibly with Joker). Straight Flush: Five consecutive cards of the same suit. High Card: Not conforming to any of above combinations.

Table 5: Out-of-domain game scripts written by human players. We skip some basic settings in the script for brevity, e.g. the number of players, bet limits.