Pokémon Battle Agent based on LLMs

Zihan Lv[1,*] Qihang Cen[1,*]

2024316064, 2024210874
[1]Tsinghua University
{lvzh24, cqh24}@mails.tsinghua.edu.cn
[*]Equal contribution

Abstract

The rapid development of LLMs has led to widespread applications in interactive environments, particularly in gaming, where LLM agents demonstrate impressive decision-making and strategy execution capabilities. We focus on developing a Pokémon battle agent based on GLMs through combining four techniques to improve agent's contextual understanding and generate effective battle commands: (1) In-context reinforcement learning; (2) Knowledge-augmented generation; (3) Consistent action generation; (4) Supervised Fine-Tuning. Through experiments on Pokémon Showdown with robots and human, we evaluate the model's win rates and strategic performance, aiming to contribute to the development of LLM agents capable of dynamic and complex environments. Results showed that despite the capability limitations of LLMs, they still demonstrated acceptable performance, proving their potential ability in analyze and strategize in the complex domain of Pokémon battles.

1 Introduction

The increasing popularity of interactive AI systems underscores the importance of developing engaging applications in gaming and education. Over the past few years, Large Language Models (LLMs) such as GPT-3, GPT-4, and other advanced architectures have revolutionized the field of natural language processing (NLP) [1] [2] [3]. These models, trained on massive datasets, demonstrate an unprecedented ability to understand and generate human-like text, enabling a wide range of applications. The capabilities of LLMs have been further amplified by advancements in prompt engineering, fine-tuning techniques, and their integration with external tools and knowledge bases.

LLM Agents is autonomous systems powered by LLMs capable of performing complex tasks through reasoning and interaction [4] [5] [6]. Its emergence has opened up new possibilities across domains. In gaming, LLM Agents offer the potential to redefine player experiences by enabling dynamic, contextually aware interactions. For example, these agents can act as virtual opponents, collaborators, or even storytellers, enhancing immersion and engagement. Similarly, in educational applications, LLM Agents can simulate interactive tutors that adapt to learners' needs, making learning experiences more personalized and effective.

The gaming industry, in particular, provides a fertile ground for the development and deployment of LLM Agents. Games often involve complex narratives, decision-making processes, and strategic interactions, which align well with the strengths of LLMs. Within this space, turn-based strategy games such as Pokémon battles present an ideal testbed for evaluating the contextual reasoning and decision-making capabilities of LLMs. Pokémon, as a well-loved franchise, not only has a rich

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

and structured gameplay system but also a global fanbase that appreciates nuanced and engaging interactions.

Despite the potential, creating effective LLM Agents for games remains a challenging task. These systems must balance contextual awareness, strategic depth, and user engagement while maintaining computational efficiency. Our project aims to address these challenges by developing a contextually aware Pokémon battle agent based on LLMs. We explore the effectiveness of four techniques to enhance the agent's performance:

(1) In-context reinforcement learning: The agent receives feedback on its previous actions in each turn. This feedback provides insights into potential errors or areas for improvement, enabling the agent to refine its decision-making process in subsequent turns.

(2) Knowledge-augmented generation: The agent's understanding of the current battle state is enriched with external knowledge, such as Pokémon type relationships and move effects. This augmentation allows the agent to make more informed and contextually relevant decisions during battles.

(3) Consistent action generation: Using prompt engineering techniques including Chain of Thought (CoT), Tree of Thought (ToT), and Self-Consistency to enhance the agent's reasoning and consistency when generating responses. These methods help the agent handle complex decision paths and improve the reliability of its outputs.

(4) Supervised Fine-Tuning: Using a dataset of gameplay actions collected from human players and high-win-rate bots to fine-tune the model to internalize effective strategies and decision patterns.

By integrating these methods, the framework ensures the agent possesses robust contextual awareness, strategic depth, and enhanced response reliability in dynamic battle scenarios. We selected the GLM model as the subject of our experiments, applying the aforementioned methods to enhance its capabilities. The enhanced model was tested within the established battle framework to observe the effectiveness of these methods and conduct evaluations about strategic performance.

2 Related Work

Generative AI and Large Language Models (LLMs) have demonstrated remarkable success in NLP tasks. A key future advancement will involve exploring how LLMs can autonomously operate in the physical world, extending their capabilities from text to action, which is crucial in the quest for Artificial General Intelligence. Games provide ideal environments for developing LLM-based agents that interact with virtual settings in ways that mimic human behavior. Below is introduction to LLM Agents in several existing types of games.

Adventure: Adventure games are known for their story-driven gameplay, where players explore environments, solve quests, and interact with characters and objects to progress the narrative. In typical adventure game like Red Dead Redemption 2, LLM-based agents could follow the main storyline and finish real missions in complex games by using GCC(General Computer Control), with minimal reliance on prior [7]. Additionally, in open-world adventure games like Minecraft incorporate crafting and exploration elements, also exists agents that could make explorations themselves [5].

Competition: Competition games challenge players to test their skills or strategies against each other to achieve victory. LLM Agents can be developed as intelligent player, offering varied levels of difficulty and strategic depth. By training the model on game-specific data, such as strategies and decision-making patterns, and applying reasoning frameworks to evaluate optimal moves, the agents could have better understanding of game state and make excellent decision. Examples include StarCraft II [8] and Pokémon Battles [9], where LLM Agents can act as formidable adversaries to enrich the player's experience.

Communication: Communication games focus on social interactions among players like conversation, challenges and controversies. LLM Agents can function as participants offering realistic reasoning and dialogue by mimic human players. This capability is achieved through fine-tuned dialogue generation and self-consistency mechanisms, allowing the agents to maintain coherent narratives and adapt their responses to specific gameplay scenarios. Examples include Werewolf [10] and Avalon [11], where LLM Agents can play the role of human-like players smoothly in the game.

Cooperation: Cooperation games emphasize players collaborating to achieve common objectives, requiring teamwork and collective problem-solving. LLM Agents can act as teammates capable of interpreting player intentions and adjusting their actions dynamically to complement the team's efforts. In games like Overcooked, LLM Agents can assist by managing resources or coordinating tasks, improving gameplay flow and minimizing frustration [12].

3 Background

3.1 Pokémon

Species: Pokémon are diverse creatures that inhabit the world, each belonging to specific species that define their characteristics and abilities. For example, *Pikachu* is known for its electric abilities, while *Charizard* is recognized for its fire-based attacks.

Type: Each Pokémon belongs to one or more types, such as Fire, Water, or Grass, influencing their strengths and weaknesses. For instance, Water-types are strong against Fire-types but weak against Electric-types.

Stats: Pokémon have distinct stats that determine their battle performance, including HP, Attack, Defense, Special Attack, Special Defense, and Speed.

Ability: Every Pokémon possesses abilities that offer passive benefits during battles. For example, the ability of *Blaziken* is *Speed Boost*, which increases its Speed each turn, enhancing its offensive potential.

Moves: Pokémon can learn various moves that they can use in battles. These moves are categorized into different types and can have varying effects, including damage, status conditions, or healing. For example, *Lucario* can learn *Aura Sphere*, a powerful Fighting-type move that never misses, while *Jigglypuff* might use *Sing* to put opponents to sleep. Trainers can strategize by selecting moves that best complement their Pokémon's types and abilities, further enhancing their effectiveness in battles.

3.2 Playground

Pokémon Showdown (https://play.pokemonshowdown.com/) is a remarkable open-source online platform that serves as a comprehensive battle simulator, enabling enthusiastic Pokémon fans to dive into the thrilling world of Pokémon battles without the necessity of engaging with the main series of Pokémon video games. Within Pokémon Showdown, there is a plethora of battle formats available, catering to a wide spectrum of preferences and skill levels.

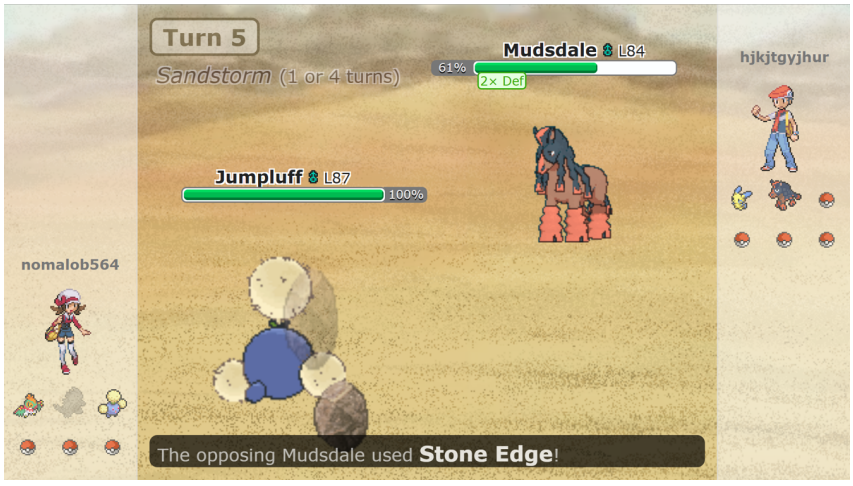

Figure 1: 2 Players Battling on Pokémon Showdown

3.3 Battle Rule

We use **[Gen 8] Random Battle** in Pokémon Showdown as our battle rule where players face off using a randomly generated team of Pokémon, all selected from the Pokémon available in Generation 8 (the Sword and Shield era). In this game mode, each player is given a random team of six Pokémon with a random moveset (often based on a general understanding of what that Pokémon can learn naturally in Generation 8). Players cannot prepare or customize their teams beforehand. Everything is determined randomly at the beginning of the match.

4 Methodology

We employed four methods to strengthen our agent. The overall framework process is as follows: in each turn, the agent receives corresponding text-based feedback on its previous to refine the decision as well as current battle state enhanced by external knowledge, such as type relationships and move effects. Then we use several prompt engineering techniques including Chain of Thought (CoT), Tree of Thought (ToT), and Self-Consistency, to improve the performance of the model's final responses. In addition, we use battle actions collected from humans and high-win-rate bots as samples to conduct supervised fine-tuning of the model, further enhancing the agent's decision-making capabilities in battles. **Figure** 2 illustrates our overall framework, which integrates four methods to enhance the capabilities of our agent.

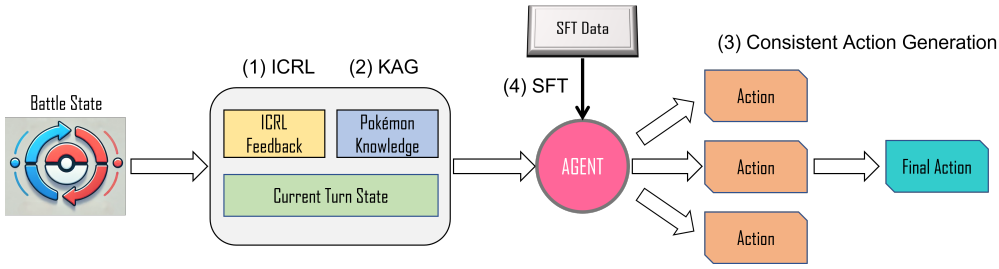

Figure 2: Overall framework

4.1 In-Context Reinforcement Learning (ICRL)

When human players making decisions, it relies on both current state and explicit or implicit feedback from previous actions, such as the effectiveness of a move used in the previous turn or any additional effects triggered by the move. But in our basic framework, the agent only receives the current battle state at each turn, which might cause the agent to repeat the same erroneous action.

For example, the Pokémon Rotom has the ability "Levitate", which grants immunity to Ground-type moves. But this information is neither included in the agent's prior knowledge nor present in the current turn's battle state (it is only displayed on the screen when player selects the move). Without feedback from previous actions, if the agent's team currently has a Ground-type Pokémon on the field, the agent may fail to understand the situation and repeatedly execute incorrect actions, ultimately leading to defeat.

In-Context Reinforcement Learning (ICRL) is a approach that leverages the language understanding capabilities of large language models (LLMs) to optimize decision-making policies. Reinforcement learning uses numerical rewards to evaluate and refine actions, while ICRL uses text-based feedback as a form of reward. By incorporating feedback from previous actions into the current context, the agent could adjust its policy. This enables the agent to learn and adapt dynamically without requiring explicit pretraining for every scenario.

In our framework, we provided the agent with four types of feedback: (1) The change in HP between turns, which reflects the actual damage caused by the attack move; (2) The effectiveness of attack move, indicating whether it's super-effective, ineffective, or immune; (3) The actual effects of executed moves, such as stat boosts or debuffs, or inflicting status conditions; (4) The execution priority of the move in the previous turn, providing a rough estimate of speed. **Figure** 3 presents an example of feedback on previous turn that includes all four types of feedback described above.

Figure 3: Feedback example of Turn 10: opposing Cramorant started Dynamax. opposing Cramorant used Max Geyser. It damaged Aegislash's HP by **39% (61% left)**. Aegislash used Close Combat. It was **ineffective** to opposing Cramorant. It damaged opposing Cramorant's HP by **18% (82% left)**. It **decreased Aegislash's def 1 level**. It **decreased Aegislash's spd 1 level**. opposing Cramorant **outspeeded** Aegislash.

4.2 Knowledge-Augmented Generation (KAG)

ICRL could reduce the agent's erroneous decisions to some extent, but it requires at least one turn of feedback samples to adjust. This delay can still lead to fatal consequences. For example, if the agent sends out a grass-type Pokémon against a fire-type Pokémon, the grass-type Pokémon may be defeated in a single turn before the agent realizes it was a poor decision.

To further mitigate such fatal decisions, additional methods are needed. Retrieval-Augmented Generation (RAG) [13] [14] has been proposed to reduce hallucinations in LLMs by enhancing the generation process with external knowledge retrieval. In our framework, we adopt a similar approach by introducing Pokémon-related knowledge data in advance, implementing a Knowledge-Augmented Generation (KAG) method.

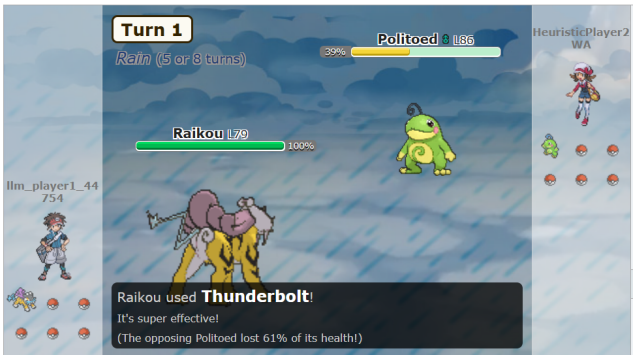

Figure 4: Extra annotation in current state: politoed as defender, **GRASS,ELECTRIC deal 2x damage; STEEL,ICE,FIRE only deal 0.5x damage to politoed**; politoed as attacker, **WATER deal 2x damage to GROUND,FIRE pokemon; WATER deal 0.5x damage to GRASS pokemon**; Move- thunderbolt: Type: ELECTRIC, Cate: Special, Power: 92, Acc: 100%, **Effect: Has a 10% chance to paralyze the target.**

We incorporate two types of external knowledge to fundamentally enhance the agent's decision-making capabilities: **type advantage/weakness relationship** and **move/ability effect**. In the original battle state, we had already annotated the type information for all Pokémon and their available moves. Bases on this, we explicitly annotated the type advantages and weaknesses for both the opponent's Pokémon and our own Pokémon. For example: *"Kartana as defender, FIGHTING deals 2x damage; NORMAL, FAIRY, STEEL, ROCK only deal 0.5x damage; GRASS only deals 0.25x damage; POISON has no effect."* Additionally, while the original state only provided the type, power and accuracy of

moves, some moves have special effects that are not within the agent's knowledge. For instance, the *Sludge* move not only deals damage but also has a 30% chance to poison the target. We explicitly annotate such effects in the state to enable the agent to utilize these additional effects to better guide its decision-making during the battle. **Figure** 4 presents an example of knowledge-augmented generation, given extra annotation to make agent use super-effective move.

4.3 Consistent Action Generation

Existing research has demonstrated that reasoning and prompting techniques can enhance the performance of LLMs in solving complex tasks. Rather than generating a single-step action, we use several prompting methods, including Chain-of-Thought (CoT) [15], Tree-of-Thought (ToT) [16] and Self-Consistency (SC) [17]. In the CoT approach, the agent first generates an intermediate reasoning process to analyze the current battle situation and then outputs an action based on that reasoning. In the ToT method, the agent produces three potential action choices and evaluates them internally to identify the optimal one. For SC, the agent generates three possible actions and selects the one with the majority vote as the final decision.

These methods play a critical role in Pokémon battles by enhancing the reasoning capabilities of large language models (LLMs), allowing the agent to make more strategic and informed decisions. CoT enables the agent to break down the complex battle situation into intermediate reasoning steps, analyzing the opponent's Pokémon, current status, and possible outcomes before determining the most appropriate action. ToT extends this approach by exploring multiple potential action paths, evaluating their consequences, and selecting the optimal action, which increases adaptability during dynamic battles. SC improves decision reliability by generating multiple candidate actions and voting for the most consistent one, reducing randomness and mitigating errors from single-shot predictions. These methods help the agent better analyze type matchups, move effects, and the evolving state of the battle, leading to more consistent, accurate, and strategic decision-making. **Figure** 5 shows the pipeline of SC, the agent generates options for 3 times and choose the majority one, which is better for the current battle state.

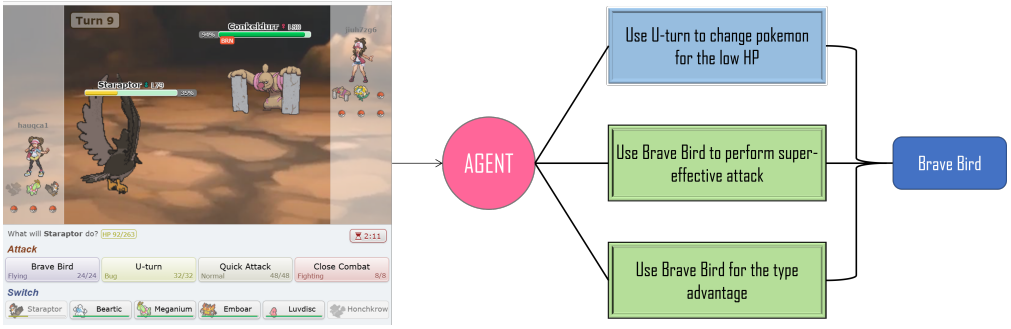

Figure 5: Agent generate 3 move options in Self-Consistency, the first one choose "U-Turn" to change pokemon while the other 2 use "Brave Bird" to cast super-effective attack. It is actually more proper to choose attack because there has an advantage both on speed and type.

4.4 Supervised Fine-Tuning (SFT)

To further enhance the agent's decision-making abilities in Pokémon battles, Supervised Fine-Tuning (SFT) [1] is employed. By collecting battle actions as training samples from human players and high-win-rate bots, the model undergoes supervised fine-tuning to align its decision-making process with optimal strategies demonstrated in these examples. We selected decisive actions from the winning side of battles, such as defeating an opponent's Pokémon or making a proper switch, as high-quality training samples. This process allows the agent to learn excellent decision patterns, such as recognizing type matchups, predicting opponent moves, and selecting the most effective actions. **Figure** 6 shows a high-quality human player battle example, human player misdirects the opponent to use dragon type attack by sending a dragon Pokémon first, and immediately change to anther one to cause immunity.

Figure 6: Human player use deceptive switch strategy to lure the opponent into making a high-powered, but immune attack.

The use of SFT would improve the agent's battle performance. It helps the model better imitate human-like strategic reasoning and adapt to diverse combat scenarios, while also reduces poor decision-making by embedding strong prior knowledge derived from expert gameplay. Fine-tuning ensures that the agent develops a more robust policy, allowing it to make consistent and reliable decisions, particularly in high-pressure or unfamiliar situations.

5 Experiment

5.1 Battle Settings

We adopt a heuristic bot to benchmark the performance of GLMs [18]. The algorithm of the bot considers the stats of Pokémon, the power of moves, and type advantages/weaknesses, uses status-boosting moves, sets entry hazards, and selects the most effective actions, which enables the bot to maximize the chances of victory and minimizing the exposure to threats. This comprehensive approach ensures that the heuristic bot provides a robust and realistic standard against which to measure the capabilities of GLMs in the complex and strategic environment of Pokémon battles.

5.2 Results and Analysis

First, we apply ICRL, KAG and consistent action generation to GLM models and evaluated their win rate against the heuristic bot in the battle rule mentioned above. Specifically, we test the performance of the CoT, ToT, and SC methods based on the application of ICRL and KAG techniques through 100 battles each method (i.e., the number of shots in the input prompt = 1, the number of turns of historical battle information = 2, and relevant knowledge is added in the prompt).

Win Rate (%)	None	CoT	ToT	SC
GLM-4-air	9.00	17.00	20.00	23.00
GLM-4v-Flash	8.00	16.00	18.00	16.00

The results show that when GLMs are directly applied to Pokémon battles without using any tuning strategies, the winning rate of the model is quite low, which is consistent with the conclusion drawn in [9]. After using ICLR, KAG, and consistent action generation, the winning rate of the models in battles has improved to some extent. However, due to the capability limitations of the base models, it still cannot reach the level comparable to that of human players.

Second, we explore the impact of SFT on GLMs in Pokémon battles. We fine-tuned a GLM-4-Flash model using battle data generated from the heuristic bot, improving the win rate of GLM-4-Flash against the heuristic bot from 2% to 10%. Meanwhile, the fine-tuned model also has some improvements in output specification, such as outputting data directly in JSON format without inclusion of the ``` `json` wrapper, and avoiding outputting illegal actions.

6 Conclusion

This study has explored the application of large language models in the context of Pokémon battles and has revealed several important findings. Firstly, when large models are applied directly to specific professional tasks without any optimization, such as Pokémon battles, their performance is significantly limited. As shown in the results, the win rate of the battles is quite low when the model is called without any enhancements. Even with the introduction of optimizations such as consistent action generation, the improvement in the model's performance is still relatively modest, which is different from the relevant results in [9]. This inconsistency may be attributed to the different capabilities of the base models. Meanwhile, we have observed a phenomenon that although many tuning strategies are applied, the strategies of the GLMs remain relatively simple and lack intelligence, such as simply switching Pokémon in order, choosing the moves that have no effect to the opponent and neglecting type advantage/weakness relationship. This reflects the problem that big models are still not capable enough to deal with complex problems.

Secondly, our research demonstrates that supervised fine-tuning of large models can effectively alter their behavioral characteristics, indicating that the behavior and performance of these models can be shaped to better suit the requirements of specific tasks or environments. By fine-tuning the model with targeted supervision, we can enhance its adaptability and effectiveness in the domain of Pokémon battles, and potentially in other specialized fields as well.

Overall, while large models hold great promise for a wide range of applications, their deployment in niche areas such as Pokémon battles requires careful optimization and fine-tuning. The limited success achieved through the application of ICLR, KAG and consistent action generation and the potential seen in supervised fine-tuning highlight the ongoing challenges and opportunities in leveraging these models for specialized tasks. Future research should focus on developing more sophisticated optimization techniques and fine-tuning strategies to further improve the performance of large models in such contexts.

References

[1] Long Ouyang, Jeff 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 Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

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

[3] OpenAI. Gpt-4 technical report, 2024.

[4] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, and Tao Gui. The rise and potential of large language model based agents: A survey, 2023.

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

[6] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6), March 2024.

[7] Weihao Tan, Wentao Zhang, Xinrun Xu, Haochong Xia, Ziluo Ding, Boyu Li, Bohan Zhou, Junpeng Yue, Jiechuan Jiang, Yewen Li, Ruyi An, Molei Qin, Chuqiao Zong, Longtao Zheng, Yujie Wu, Xiaoqiang Chai, Yifei Bi, Tianbao Xie, Pengjie Gu, Xiyun Li, Ceyao Zhang, Long Tian, Chaojie Wang, Xinrun Wang, Börje F. Karlsson, Bo An, Shuicheng Yan, and Zongqing Lu. Cradle: Empowering foundation agents towards general computer control, 2024.

[8] Weiyu Ma, Qirui Mi, Yongcheng Zeng, Xue Yan, Yuqiao Wu, Runji Lin, Haifeng Zhang, and Jun Wang. Large language models play starcraft ii: Benchmarks and a chain of summarization approach, 2024.

[9] Sihao Hu, Tiansheng Huang, and Ling Liu. Pokellmon: A human-parity agent for pokemon battles with large language models, 2024.

[10] Yuzhuang Xu, Shuo Wang, Peng Li, Fuwen Luo, Xiaolong Wang, Weidong Liu, and Yang Liu. Exploring large language models for communication games: An empirical study on werewolf, 2024.

[11] Jonathan Light, Min Cai, Sheng Shen, and Ziniu Hu. From text to tactic: Evaluating LLMs playing the game of avalon. In *NeurIPS 2023 Foundation Models for Decision Making Workshop*, 2023.

[12] Micah Carroll, Rohin Shah, Mark K. Ho, Thomas L. Griffiths, Sanjit A. Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination, 2020.

[13] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021.

[14] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-augmented language model pre-training, 2020.

[15] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

[16] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023.

[17] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023.

[18] H. Sahovic. poke-env: Heuristicbot. `https://github.com/hsahovic/poke-env/blob/master/src/poke_env/player/baselines.py`, 2023. Accessed: 2024-12-18.