RL- and LLM-based AI Solvers for the Game of Wordle/Fibble

Krerkkiat Chusap
School of EECS
Ohio University
Athens, Ohio, USA
kc555014@ohio.edu

Colin Murphy
School of EECS
Ohio University
Athens, Ohio, USA
cm787623@ohio.edu

Clay Hess
School of EECS
Ohio University
Athens, Ohio, USA
ch559521@ohio.edu

Sai Deepa Kadaru
School of EECS
Ohio University
Athens, Ohio, USA
sk414623@ohio.edu

Rohit Buccapatnam
School of EECS
Ohio University
Athens, Ohio, USA
rb064323@ohio.edu

Chang Liu
School of Electrical Engineering and Computer Science
Ohio University
Athens, Ohio, USA
liuc@ohio.edu

***Abstract*—This paper investigates the performance of reinforcement learning (RL)-based AI agents and large language model (LLM)-based AI agents in tackling the word-guessing game Wordle and its variant, Fibble, which introduces misleading feedback. We conducted experiments to compare a range of RL-based approaches and several LLM chatbots, revealing distinct performance differences, especially when confronted with deceptive clues. Our findings show that an RL algorithm specifically designed to handle misleading information outperformed others in Fibble, while a standard RL algorithm designed for the original Wordle excelled in that game but struggled with Fibble. LLM-based solvers demonstrated reasonable performance across both games but generally fell short of the effectiveness of RL-based solvers.**

***Index Terms*—reinforcement learning, LLM Chatbots, game AI agents**

I. Introduction

Reinforcement learning [1] is a popular and effective machine learning method to train game AI agents and solvers. It is primarily based on experiences instead of existing data. Its success in helping game AI agents defeat human masters of the games of Go (2016) [2], Dota 2 (2019) [3] and StarCraft II (2019) [4] were major milestones in the history of game AI.

Large Language Models (LLMs) was an outcome of decades of research of the Natural Language Processing community. More recently, transformers (2017) [5] and BERT (Bidirectional Encoder Representations from Transformers) (2019) [6] from Google helped NLP approaches to be more scalable. Open AI's Reinforcement Learning from Human Feedback (RLHF) (2017) [7], particularly using Proximal Policy Optimization (PPO) (2017) [8], as seen in models like ChatGPT, played a crucial role in aligning LLMs with human preferences [9] and pushed LLM chatbots beyond the traditional text-based domains to become general problem-solving solutions, although their effectiveness in this area is still largely unproven.

In this paper, we study the effectiveness of RL-based approaches as well as LLM-based methods as AI solvers for the game of Wordle and Fibble.

In the game of Wordle [10], a player must guess a secret five-letter word within six attempts. Feedback is provided accurately in the form of five colored blocks with green representing the guessed letter is correct and in the right position, yellow meaning the guessed letter is correct but in the wrong position, and gray meaning the guessed letter is incorrect and not in the secret word.

In the game of Fibble [11], on the other hand, feedback could be a lie. In Fibble, one needs to guess a secret five-letter word, similar to Wordle. Feedback is also provided in five blocks of three different colors for each guess. Exactly one of these feedback blocks is a lie, though. This makes Fibble a lot harder to win than Wordle, even if more attempts are allowed. In original Fibble, one has nine guesses with the first one already provided.

There are effective information-theory-based solvers for Wordle that can maximize the amount of information gained with each guess. For this type of game, that is likely an optimal solution. Regardless, it is interesting to see how RL type of approaches work for Wordle/Fibble and how they compare to LLM-based solutions.

One particular question we aim to answer is how the presence of misleading clues impacts the different types of AI agents that are currently considered the best-performing ones in their categories? We designed and ran experiments to find out perhaps not the complete answer yet but at least some initial clues.

II. Related Work

A. RL-based Game AI Agents

Reinforcement learning [1] algorithms typically are designed to rely on both exploration and exploitation and do not require a perfect model of the environment. Therefore, they can naturally defend against certain amounts of false

information present in the environment. Robustness, a property of RL algorithms, refers to "the ability to cope with variations or uncertainty of one's environment." [12] For example, there may be subtle differences between the real environment and the simulated model used for training, which is called the sim-to-real gap. In this paper, our focus is not on this type of unintended gap that is often the result of trade-offs between model accuracy and performance. Rather, we are interested in intentionally misleading clues that are part of the challenges of a game. Out of the four types of robustness of RL algorithms described in [12], namely transition robustness, disturbance robustness, action robustness, and observation robustness, this paper addresses primarily observation robustness of RL algorithms.

To make RL algorithms more robust to an adversarial attack, Zhang et al. proposed a state-adversarial Markov decision process [13]. The attack in question is similar to the game with misleading clues that we are exploring since the model's observation of the state of the environment is modified while the true state of the environment is kept unchanged.

Intentional adversarial attacks on deep reinforcement learning algorithms could have critical real-world implications when safety-critical applications such as autonomous driving are involved. Ilahi et al. classified adversarial attacks on deep reinforcement learning into four categories involving the state space, the reward function, the action space, and the model space [14]. All such attacks are modifications to part of the information accessible to the RL algorithms. Their study, however, is broader and not focused on games. Our work to measure the practical impact of misleading clues in a much more limited scope of a few selected games could help provide additional insights into the context of games.

B. Large Language Models (LLMs)

Recently, LLM chatbots have been successfully used in solving a variety of problems. In game development and game AI research, there are various works that study the efficacy of LLM chatbots in game-playing AIs, game difficulty evaluation framework, and many more. However, few works are studying the potential of LLM chatbots as a sidekick for players.

An LLM agent was used to measure the difficulty of Wordle and other games [15]. The developed framework used the number of steps the agent took as a measurement of the difficulty of the game under investigation. The authors suggested that the LLM agent produces an evaluation that is similar to the hand-built rule-based model. The findings concluded that the LLM agent has the potential as a platform to evaluate the difficulty of games.

Wang et al. leveraged an LLM chatbot's capability for natural language interaction to create an application that teaches K-12 students the concepts of artificial intelligence (AI) [16]. The authors achieved the teaching by allowing students to use an LLM chatbot to create a "heuristic" for an AI agent to play a strategy game.

Hu et al. showed that LLMs are being studied in various types of games ranging from adventure, communication, competition, cooperation, simulation, and crafting/exploration [17]. For example, Hu et al. presented a paper from Carroll et al that explored the the possibility of using an LLM agent as a sidekick in cooperative games like Overcooked [18]. However, less work is being put into exploring LLMs and their robustness against lies.

The remainder of this paper is structured as follows. Section III presents the results of a standard RL-based AI solver applied to the Wordle/Fibble games. Section IV follows with a discussion of a specialized RL-based AI solver we developed to handle potentially misleading clues. Finally, Section V introduces our LLM-based solvers for the Wordle/Fibble games.

III. RL-based Wordle/Fibble Game AI Solvers

Wordle [10] is a game to guess a 5-letter secret word with 6 attempts. All feedback is true. No lies are involved. Fibble is a game in which the player guesses a 5-letter secret word in a similar way, except there is exactly one lie in the feedback for each guess. We refer to the game without lies as Wordle or Fibble$_0$. We refer to the original game of Fibble [11] with one lie per feedback line as Fibble or Fibble$_1$. We refer to the modified version of the Fibble game with 2, 3, 4, or 5 lies per feedback line as Fibble$_2$, Fibble$_3$, Fibble$_4$, or Fibble$_5$.

We started with RL-based game AI solvers designed just for Wordle without consideration for lies. Based on the game of Wordle, a naive RL-based solver would define the observation space as a 5x6 array of letters in color. The action space would be an array of five elements, each with 26 possibilities (26 letters). This action space would be too large to search (26^5, almost 12 million) and most of the actions in this space would be invalid (not a valid word). Instead, we used an optimized RL-based game AI for Wordle from [19].

The game environment can select one secret word from a list of 2,315 words. The model, however, has a larger word list (12,972 words) to select from, which includes the words in the first list.

A. Observation space for the optimized RL Wordle solver

The observation space for this Wordle solver consists of several components:

- **GuessesLeft**: The number of remaining guesses the agent has before the game ends. This is a dynamic component of the state that decreases with each guess the agent makes.
- **IsKnown**: A binary vector of length 26 (for each letter of the alphabet) that indicates either the presence of the letter in any of the guessed words or the absence of the letter in all of the guessed words.
- **IsIn**: Another binary vector of length 26, indicates for each letter of the alphabet whether it is present in the answer. This does not provide positional information but helps the agent track which letters are part of the answer.
- **Coloring**: A 3-dimensional array with dimensions 26 (number of letters) x 5 (number of positions in the word) x 2 (binary indicators for Green/Yellow or Red where Red

is equivalent to Gray which represents the letter not in the target word). This array captures detailed information about each letter's presence and positional correctness in the target word. For each letter and position, it indicates whether the letter is definitely not in that position, or if it is correctly placed.

B. Action Space for the optimized RL Wordle solver

The action space comprises all possible actions that an agent can undertake at any given point in time. In traditional approaches, this includes all possible actions, resulting in a complex action space. To reduce its complexity, the action space is structured into independent actions (selecting a word decomposes into selecting letters for specific positions). This is achieved using a One-Hot Vector, where each word in our vocabulary is represented as a 130-value vector (26×5). This vector is divided into five segments, one for each of the five positions in a word, with each segment containing a 26-component vector representing the one-hot encoded letter at that position. Rather than choosing from thousands of potential words, the network now selects among a fixed set of 26 letters for five distinct positions. This method simplifies the learning process to navigate the action space effectively.

C. Reward Function for the optimized RL Wordle solver

For each attempt, the agent receives the following rewards: +1 for each correctly guessed green and yellow letter, a penalty of -5 for incorrect guesses, a substantial reward of +15 in case of successfully solving the Wordle, a significant penalty of -15 in case of an unsuccessful 6th attempt resulting in the loss of the game, and a minor penalty of -0.1 for guessing a previously guessed green letter in the same position. These reward values are designed to make accurate and efficient guesses, encouraging successful Wordle solutions while discouraging incorrect entries and redundant guesses.

D. Results

We evaluated a reinforcement learning (RL) model (DQN) on Wordle. The original Wordle game allows six attempts, so we limited our evaluation on Wordle and on all Fibble variants to six guesses per game. Although the original Fibble rules permit nine attempts, we constrained Fibble to six guesses in order to maintain a fair, one-to-one comparison. The goal was to train an agent that could not only play the game with high proficiency but also adapt its strategy to optimize its win rate. The model was subjected to a total of 9 million training episodes, with an initial warm-up phase of 100 episodes to allow for initial exploration without significant policy updates. An epsilon-greedy strategy was employed with epsilon starting at 1 (100% exploration) and decaying to 0.05 (5% exploration) at a decay rate of 0.9954, ensuring a gradual shift from exploration to exploitation. The learning rate (alpha) was set at 0.4, with a replay buffer size of 100,000 to store and reuse past experiences, facilitating more efficient learning. The agent was trained across 8 parallel environments, with optimization occurring every 8 steps to improve learning efficiency and

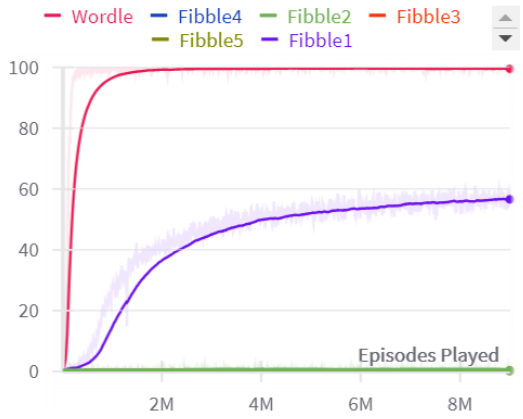

Fig. 1. Win rate comparisons among Wordle and Fibble.

stability. The performance metrics indicate that the RL model achieved a remarkable win rate of 99.8% after 50 minutes of training. This high level of proficiency underscores the effectiveness of the chosen architecture, training parameters, and reinforcement learning strategy in mastering the Wordle game.

We trained the same RL model for Fibble for 9 million episodes to expose it to the variant's deceptive feedback. The model achieved a win rate of 56.82%. This performance is significantly lower than the win rate for the standard Wordle game. When more than one lies were introduced into the Wordle game, the win rates further decreased. The win rate for Fibble_2 was 0.43%. For Fibble_3, it decreased to 0.27%. For Fibble_4, it further reduced to 0.16%. For Fibble_5, it was slightly higher at 0.29%. Figures 1 and 2 show the change in win rates over the entire training process. All win rate results are listed in Table IV for comparison with another algorithm specifically designed for Fibble.

The results of the Wordle game with lies indicate the heightened difficulty and complexity introduced by the inclusion of misleading information. The training parameters, including the action space and the observation space, remained the same, utilizing an epsilon-greedy strategy to maintain a balance between exploration and exploitation.

IV. A Targeted RL-based Solver for Fibble

In addition to measuring how robust an RL-based Wordle solver was when applied to Fibble, we were also interested in learning if an RL solver specifically designed for Fibble could perform better. For this purpose, we implemented an RL-based game AI for Wordle (with no lies) and Fibble (with one or more lies) with specific consideration targeting the possibility of lies. All games allow a total of eight guesses. In the case of Fibble, one random word is used as the initial guess.

A. Data preparation for the targeted RL Fibble solver

The first step was to obtain a list of 5,757 five-letter words that are considered legal guesses in the game of Wordle. The second step was to compute the probabilities of each letter

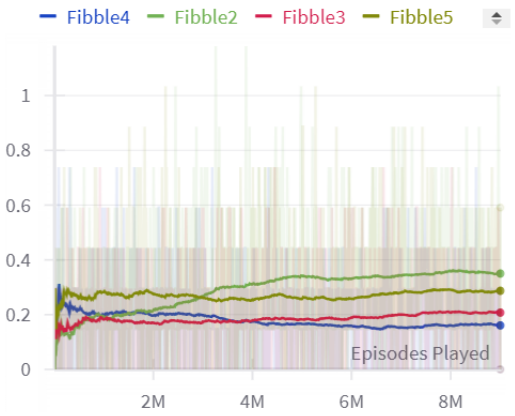

Fig. 2. A zoomed-in view of the bottom of Fig. 1 for Fibble 2-5.

showing up at each position (Positions 1-5). The results are shown in Table I. For example, the value 0.0514 in the topmost cell means the letter A has a 5.14% chance of showing up in the 1st position of a 5-letter word.

TABLE I
LETTER PROBABILITY OF 5-LETTER WORDS IN WORDLE/FIBBLE

	1	2	3	4	5
A	0.0514	0.1615	0.1051	0.0589	0.0309
B	0.0750	0.0056	0.0222	0.0172	0.0042
C	0.0764	0.0142	0.0320	0.0365	0.0083
D	0.0540	0.0075	0.0309	0.0379	0.0749
E	0.0224	0.1146	0.0690	0.2133	0.1034
F	0.0552	0.0021	0.0151	0.0174	0.0076
G	0.0485	0.0042	0.0241	0.0306	0.0106
H	0.0415	0.0471	0.0068	0.0127	0.0334
I	0.0129	0.1169	0.0896	0.0493	0.0078
J	0.0127	0.0007	0.0014	0.0007	0.0000
K	0.0158	0.0050	0.0156	0.0422	0.0248
L	0.0471	0.0625	0.0674	0.0634	0.0351
M	0.0518	0.0123	0.0363	0.0327	0.0134
N	0.0205	0.0292	0.0712	0.0670	0.0353
O	0.0188	0.1582	0.0841	0.0455	0.0261
P	0.0670	0.0196	0.0294	0.0340	0.0158
Q	0.0068	0.0017	0.0007	0.0000	0.0000
R	0.0466	0.0792	0.0825	0.0538	0.0697
S	0.1258	0.0069	0.0431	0.0446	0.3064
T	0.0653	0.0212	0.0486	0.0776	0.0625
U	0.0130	0.0928	0.0544	0.0268	0.0023
V	0.0189	0.0047	0.0210	0.0106	0.0000
W	0.0396	0.0141	0.0170	0.0122	0.0049
X	0.0007	0.0057	0.0116	0.0009	0.0052
Y	0.0082	0.0113	0.0118	0.0071	0.1155
Z	0.0042	0.0010	0.0090	0.0071	0.0021
Total	1.0000	1.0000	1.0000	1.0000	1.0000

These five lists are loaded in five dictionaries representing letter probabilities at each index of 5-letter words. Each dictionary contains probabilities summing up to 1.

B. Observation space of the targeted RL Fibble solver

The observation space is a dictionary of two arrays, with discrete values ranging from -1 to 2. The "board" is a two-dimensional 9×5 array representing the board's colors. (Only colors, not letters.) The "alphabet" is a one-dimensional 26×1 array representing the last color of each letter. In each of these arrays, the value -1 represents an unguessed space/letter, 0 for gray, 1 for yellow, and 2 for green. This way, the model can observe the color-coded feedback of its guesses. An example of the state of the game is shown in Fig. 3 and the representation of this game is shown in Fig. 4.

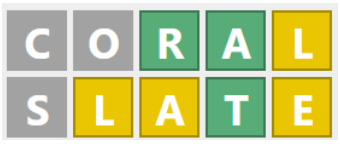

Fig. 3. An example with the first two guesses of the game of Wordle.

0	0	2	2	1
0	1	1	2	1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1

A	B	C	D	E	...	S	T	...	Y	Z
2	-1	0	-1	1	...	0	2	...	-1	-1

Fig. 4. A representation of the board (top) and the alphabet (bottom) after two guesses.

C. Action space of the targeted RL Fibble solver

The action space currently contains five real numbers, namely $GreenWeight$, $YellowWeight$, $GrayWeight$, $GaussianWeight$, and $DictionaryWeight$. The first three can range from 1/16 to 16. This range is arbitrarily chosen with no particular meaning. These parameters allow the model to guess by scoring words with letters that previously have been green, yellow, or gray in the game of Fibble. $GaussianWeight$ and $DictionaryWeight$ are floats from 0.0 to 1.0. The latter functions as a boolean value. If $DictionaryWeight$ is less than 0.5, the model scores each word in the entire dictionary. Should the value be greater than or equal to 0.5, the model will only score words that are possible solutions given the existing guesses, their colors, and the known number of lies per guess. The $GaussianWeight$ allows the model to be biased for letters that scored very high or very low.

D. The reward function of the targeted Fibble solver

Games in progress receive a reward of (Number of green letters in last 3 guesses + 0.3 $\times$ the number of yellow letters in last 3 guesses), divided by 5 $\times$ minimum(3, total # of guesses made) (see Equation 1). In other words, the denominator is 5, 10, or 15. The minimum value of the numerator is 0 (all grays). The rewards for games in progress are bounded by [0,1].

Games won receive rewards of one plus the number of remaining guesses. Lost games receive rewards of -1. These are shown as Equation 2 and Equation 3 respectively.

$$Reward = \frac{(\#\ green\ letters) + (0.3 \times \#\ yellow\ letters)}{min(15, 5 \times (\#\ of\ guesses))} \quad (1)$$

$$Winning\ Reward = 1 + (\#\ of\ remaining\ guesses) \quad (2)$$

$$Losing\ Reward = -1 \quad (3)$$

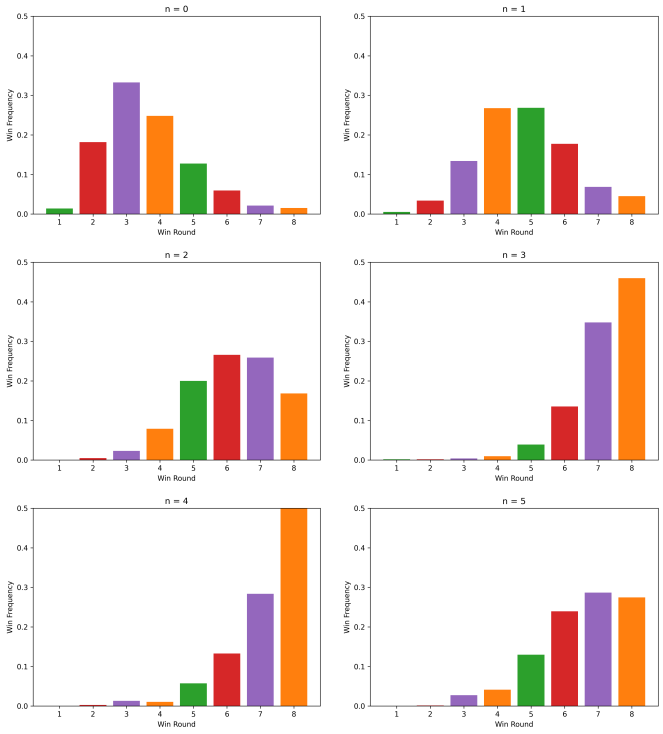

Fig. 5. Wins by the number of rounds for Wordle (top left), $Fibble_1$ (top right), $Fibble_2$ (middle left) to $Fibble_5$ (bottom right).

E. An illustrative example of the RL process

This example shows a moment in the RL training of the targeted Fibble solver. At this moment, the model weights have the values as listed in Table II, after two guesses have been made as shown in Fig. 3.

TABLE II
WEIGHTS FOR EACH TYPE OF THE LETTER

Green Weight	Yellow Weight	Gray Weight	Gaussian Weight	Dictionary Weight
4	2	0.5	0.8	0.6

The letter probability table (a 5×26 matrix of floats), as shown in Table I, is used to process each letter in Fig. 3.

Considering the first guess of the game board in Fig 3, the probability for the first letter "C", according to the Table I, is 0.0764. We multiply this with the corresponding weight (gray in this case), which is 0.5, (see Table II), to become 0.0382. It should be noted the weights for "C" in columns 2-5 remain unchanged.

"O" in the second column of the game board is also gray. Its second column weight is updated from 0.158 to 0.079. The "O" values in columns 1,3,4,5 remain unchanged.

"R" in the third column is green. We multiply its 3rd column value by 4, the Green Weight. This process is continued for each letter until all letters in all guesses are processed.

Yellows are treated differently. Assuming there are no lies, "L" could not be in the 5th position. Because of this, we multiply the "L" probability in the 5th position by the Gray Weight. "L" in Positions 1,2,3, and 4 will be updated by the Yellow Weight.

This continues until every letter has been processed.

At the end of this process, each column is normalized so the sum of the weights is 1, with the intention of the weights behaving like probabilities.

Since the Dictionary Weight is greater than or equal to 0.5, we only use words that could be possible solutions. This narrows the word list all the way only to seven words, which are 'early', 'delay', 'relax', 'relay', 'belay', 'velar', and 'feral'.

We then iterate each word in the word list, that is both 'coral' and 'slate', and use the Gaussian Weight to give the word an overall weight. For each letter, we will take the letter's weight in its respective column, and calculate a new value from a normal curve. For example, let us consider the possible solution 'Early'. "E" has a weight of 0.0402 in the first column (after the column is normalized to sum to 1). We will take this number and calculate its probability density on a Gaussian curve with a mean of 0.8, and a standard deviation of 0.1, which gives us 0. We will calculate this value for each letter and compute the sum for each word. The result of doing this for each possible word is shown in Table III.

TABLE III
WEIGHT OF THE REMAINING POSSIBLE SOLUTIONS

Word	Weight
early	34.2050×10^{-7}
feral	6.5231×10^{-7}
belay	4.0990×10^{-7}
delay	4.0989×10^{-7}
relay	4.0989×10^{-7}
velar	4.0959×10^{-7}
relax	4.0958×10^{-7}

Finally, we sort the word list by weight, from highest to lowest. The first word in the list that hasn't already been guessed will be guessed. In this case, 'Early' Will be the next word guessed.

The current models are PPO models [8] trained on 100,000 global time steps. The number 100K was picked because, beyond that, there was not much progress in the mean iteration reward. Stable-baselines-3 provided the RL implementation, and our environment was normalized with Gymnasium [20], [21].

F. Results

As indicated by the results for the RL-based AI solver for Fibble in IV, it was clear that as the number of lies went up (from Wordle to $Fibble_4$), the win rate decreased as the problems became harder to resolve. A surprising yet logical

TABLE IV
WORDLE AND FIBBLE RL-BASED AI SOLVER RESULTS

	Wordle	Fibble	$Fibble_2$	$Fibble_3$	$Fibble_4$	$Fibble_5$
RL-based AI Solver for Fibble (9 guesses)						
Win%	99.40%	96.90%	84.50%	49.0%	38.60%	61.1%
Mean	3.67	4.73	6.17	7.13	7.10	6.46
LLM-based AI Solver for Fibble (gpt-4o; 9 guesses)						
Win%	21.27%	14.89%	2.12%	0.00%	0.00%	0.00%
Mean	5.78	8.76	8.97	9.00	9.00	9.00
RL-based AI Solver for Wordle (6 guesses)						
Win%	99.80%	56.82%	0.43%	0.27%	0.16%	0.29%

discovery was that the win rate for $Fibble_5$ was actually higher than $Fibble_4$ because when the feedback for all 5 letters in the guess were lies, one did not have to guess which ones were lies. This actually made the game easier to play. The distribution of wins by rounds for Wordle and Fibble 1 through 5 are listed in Fig. 5.

Results from the Wordle/Fibble experiments show that RL algorithms indeed are naturally more robust in the presence of misleading information, targeted algorithm design with specific considerations for lies can significantly further improve win rates.

V. AN LLM-BASED SOLVER FOR WORDLE/FIBBLE

This study investigates the ability of large language models (LLMs) to play the word-guessing game, Wordle, effectively. The experiment was designed to assess how well models such as GPT-4, GPT-4o, and o1-preview perform when tasked with selecting five-letter words based on feedback from previous guesses. The feedback system follows the standard Wordle format, where each letter in a guessed word is marked as either:

- Incorrect (letter is not in the target word)
- Present (letter is in the word but in the wrong position)
- Correct Place (letter is in the correct position)

In order to test a lessor-known game, a small modification will be applied to the Wordle game. The feedback in one position will be set to the wrong color and the number of attempts will be set to nine. This game is called Fibble. In addition to being not as popular as Wordle, this game also has some uncertainty that the LLM will have to navigate.

A. Testing procedures

To efficiently get the LLM to make relevant guesses, effective prompting has to occur. In the case of these tests, three main prompts were provided to the LLM:

1) A basic prompt telling the LLM what type of game it is playing and the rules by which the game goes.
2) The words that were guessed previously, along with the feedback associated with them.
3) The number of tries remaining (in an attempt to give the model a sense of urgency).
4) When testing Fibble, the additional rules will be given with the number of lies that the game will have, in this case just one.

Each iteration was evaluated for 47 games[1], with the average success rate and number of tries recorded in Figures 6 and 7 respectively. Note that only 27 games were evaluated for Fibble with model o1-preview due to budgetary constraints. See Appendix A for sample prompts.

B. Results

TABLE V
WORDLE AND FIBBLE LLM-BASED AI SOLVER RESULTS

Model	Win%	Mean Tries
o1-preview	95.74%	3.89
gpt-4o-5-retries	59.57%	5.02
gpt-4o	21.27%	5.78
gpt-4	14.89%	5.89
o1-preview-fibble	29.62%	8.48

In the first phase of testing, GPT-4 and GPT-4o were evaluated with strict adherence to the feedback provided. The models were not allowed to make corrections if their guesses contradicted the feedback given. Under these constraints, GPT-4 and GPT-4o exhibited a relatively low success rate, with win rates averaging around 20% (14.89% and 21.27% respectively). This suggests that while LLMs are capable of making reasonable word choices, they struggle with maintaining consistency in applying feedback constraints within a single iteration. However, o1-preview scored a surprisingly high score of 95.74% win rate with the average number of tries hanging just below four, suggesting that more highly trained models exponentially increase their knowledge of word guessing games.

To investigate whether performance could be improved for GPT-4o, a second phase of testing was introduced in which models were given additional opportunities to refine their guesses. If the model suggested a word that did not adhere to prior feedback, it was given another opportunity to adjust its selection. Additionally, a maximum number of retry attempts must be set in order to prevent the LLM from recursively guessing forever. For this case, five is a good starting point. Under these conditions, the win rate increased significantly to around 60% and the average number of tries dropped to around five. This indicates that while LLMs struggle to enforce strict logical constraints in a single-step prediction, they perform much better when allowed multiple attempts to self-correct.

To study the effect of lies in the feedback in the Fibble game on the LLM-based agent. We use GPT-4o to solve Fibble and its variant ($Fibble_1$ to $Fibble_5$). As shown in Table IV, the model wins, on average, at 14.89% at the Fibble game. The win rate falls down to 2.12% on $Fibble_2$ and down to 0.00% for the rest of Fibble's variants. Out of curiosity, but with budgetary constraints, we explore the o1-preview's performance on the Fibble game. The model has a win rate of 29.63% with an average number of tries at 8.48 (see Table V). This suggests that even in a less familiar game with deceptive

[1]We evaluated all models for 50 games, but due to OpenAI API's limitation, the result for Wordle game with o1-preview model is inconclusive after the 47th game.

feedback, the LLM still manages to extract enough reliable information to formulate effective guesses. However, when the number of lies increases, the LLM's performance quickly deteriorates.

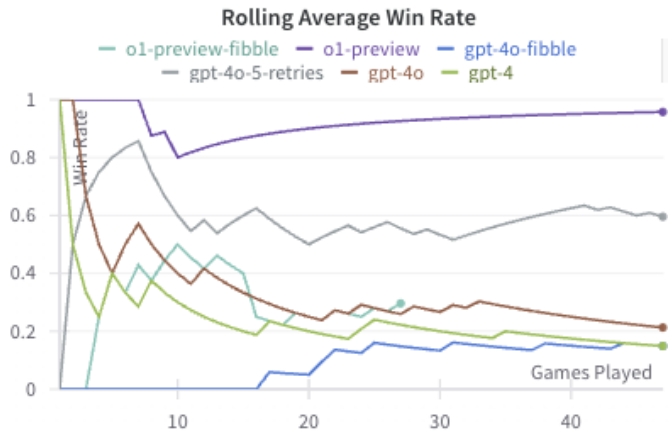

Fig. 6. This figure presents the average success rates of GPT-4 and GPT-4o in both Wordle and Fibble over 47 trials. When not marked, the result is for the Wordle game. For Wordle, GPT-4o has an average win rate of 14.89%. GPT-4o only performs slightly better at a 21.27% win rate. However, when GPT-4o is allowed to re-prompt with an external constraints checker, its win rate is increased to 59.57%. Lastly, o1-preview performs the best in a Wordle game at a win rate of 95.74%. For Fibble, GPT-4o has a win rate of 14.89%. Due to budgetary constraints, we can only evaluate the o1-preview for the Fibble game for 27 games. It has a win rate of 29.62%.

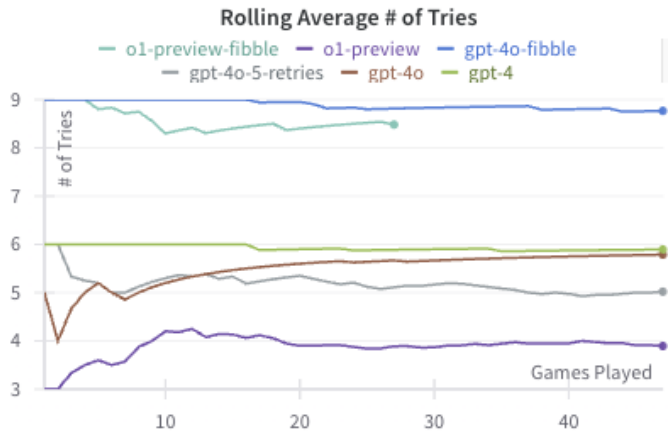

Fig. 7. This figure illustrates the average number of attempts taken per successful Wordle game by the LLMs. Under strict adherence to feedback, models required close to the maximum six tries, while iterative refinement reduced this to around 4-5 attempts per win. In Fibble, the added uncertainty slightly increased the number of required tries, though the model still managed to extract enough reliable information to succeed in select cases. The comparison underscores the importance of structured correction in improving LLM efficiency in word-guessing games.

VI. Discussion and Future Work

For the Wordle game, the RL-based trained specifically for Wordle has the best performance at a 99.80% win rate. While the older GPT models (GPT-4o and GPT-4) show less than a 50% win rate, the new o1-preview shows a comparable performance at a 95.74% win rate. This appears to be the result of the chain of thought prompting since when we allow GPT-4o to re-prompt with just a simple external constraint checker, the model's win rate increases to 59.57%.

For the Fibble game, the RL-based model designed around the presence of a lie performs the best at a 96.90% win rate. While not being trained specifically for Fibble and is at a disadvantage of six guesses, the Wordle-RL model still beat the LLM-based model at a 56.82% win rate. The LLM-based model only archives a 14.89% win rate. We suspect that the LLM relies on the existing concept for Wordle and fails to generalize to account for the lie in Fibble. This is further confirmed when the LLM-based model fails (with a 0.00% win rate) at Fibble with three or more lies in the feedback while the Fibble-RL can leverage facts that all feedback is lie and perform better in Fibble_5 at 61.10%

The findings suggest that when a game is discussed in the literature and thus exists in the training data for LLMs, they perform better. Their ability to solve Wordle and its variants is significantly enhanced when they receive guidance, such as being allowed to re-attempt guesses that contradict prior feedback in [22]. However, LLMs appear to have trouble handling the lie in the Fibble game and are unable to play effectively when there are three or more lies in the game.

More work is needed to thoroughly investigate other LLMs. More study is also needed to discover whether or not the LLMs pick up on the important of the feedback.

VII. Summary

To find out if RL models and LLM-based agents have intrinsic robustness against lies in a word game (Fibble, and Fibble's variants), we implement two RL agents and one LLM-based agent. The first RL agent is trained only on the original Wordle game while the second RL agent is trained on the Fibble game and is designed around the presence of lies in the feedback.

For the LLM agent, we explore GPT-4o's capability to act as a game-playing AI agent for Wordle. We conduct limited experiments for the LLM-based model's robustness against lies in Fibble (and its variants). For comparison, we ran an even more limited experiment using the new o1-preview model for the original Fibble game.

Thus, the new LLM model can act as a game-playing AI agent for Wordle, but it is shackled by its existing knowledge and fails to derive the existence of lies from the feedback in the Fibble game. Overall, the RL-based solver performs better at a word game and a word game with lies in the feedback while LLM struggles. Further studies are needed to explore the newer model's robustness against lies in word games.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[3] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

[4] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[5] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 2019, pp. 4171–4186.

[7] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," *Advances in neural information processing systems*, vol. 30, 2017.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[9] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022.

[10] "The game of Wordle." [Online]. Available: https://www.nytimes.com/games/wordle/index.html

[11] "The game of Fibble." [Online]. Available: https://fibble.xyz/

[12] J. Moos, K. Hansel, H. Abdulsamad, S. Stark, D. Clever, and J. Peters, "Robust reinforcement learning: A review of foundations and recent advances," *Machine Learning and Knowledge Extraction*, vol. 4, no. 1, pp. 276–315, 2022.

[13] H. Zhang, H. Chen, C. Xiao, B. Li, M. Liu, D. Boning, and C.-J. Hsieh, "Robust deep reinforcement learning against adversarial perturbations on state observations," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 21 024–21 037.

[14] I. Ilahi, M. Usama, J. Qadir, M. U. Janjua, A. Al-Fuqaha, D. T. Hoang, and D. Niyato, "Challenges and countermeasures for adversarial attacks on deep reinforcement learning," *IEEE Transactions on Artificial Intelligence*, vol. 3, no. 2, pp. 90–109, 2021.

[15] C. Xiao and B. Z. Yang, "LLMs may not be human-level players, but they can be testers: Measuring game difficulty with llm agents," 2024. [Online]. Available: https://arxiv.org/abs/2410.02829

[16] Y. Wang, S. Guo, L. Ling, and C. W. Tan, "Nemobot: Crafting strategic gaming llm agents for k-12 ai education," in *Proceedings of the Eleventh ACM Conference on Learning@ Scale*, 2024, pp. 393–397.

[17] S. Hu, T. Huang, F. Ilhan, S. Tekin, G. Liu, R. Kompella, and L. Liu, "A survey on large language model-based game agents," *arXiv preprint arXiv:2404.02039*, 2024.

[18] M. Carroll, R. Shah, M. K. Ho, T. Griffiths, S. Seshia, P. Abbeel, and A. Dragan, "On the utility of learning about humans for human-ai coordination," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/f5b1b89d98b7286673128a5fb112cb9a-Paper.pdf

[19] Voorhs, "Implementing a wordle solver with reinforcement learning," 2023. [Online]. Available: https://github.com/voorhs/wordle-rl

[20] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[21] M. Towers, A. Kwiatkowski, J. Terry, J. U. Balis, G. De Cola, T. Deleu, M. Goulão, A. Kallinteris, M. Krimmel, A. KG *et al.*, "Gymnasium: A standard interface for reinforcement learning environments," *arXiv preprint arXiv:2407.17032*, 2024.

[22] T. Liang, Z. He, J. Huang, W. Wang, W. Jiao, R. Wang, Y. Yang, Z. Tu, S. Shi, and X. Wang, "Leveraging word guessing games to assess the intelligence of large language models," 2023. [Online]. Available: https://arxiv.org/abs/2310.20499

APPENDIX A
PROMPTS FOR WORDLE/FIBBLE

```
1 You will guess a 5 letter word based off
      previous guesses and feedback in the form
      of correct: correct place, present: letter
      is present but not in correct spot, and
      incorrect: letter is not present in the
      word.
2 There may be more than one lies, meaning one or
       more feedbacks will be incorrect.
3 Assume there are no lies unless otherwise
      stated.
4 Respond with the 5 letter word and then the
      reason why you chose that word, nothing
      else!
5 If no feedback is provided, you must guess the
      word without feedback.
6 You only have a certain amount of tries to get
      the word.
```

Listing 1. Initial rules prompt for Wordle/Fibble.

```
1 Guess: ARISE
2 Feedback:
3 A: incorrect
4 R: incorrect
5 I: correct
6 S: present
7 E: present
8 You have 5 tries remaining.
```

Listing 2. Example word feedback prompt for Wordle/Fibble.

```
1 There is one lie present in the game
```

Listing 3. Number of lies prompt for Fibble

```
1 'E' is not a possible letter for this spot,
      valid letters are: QWTYUOPADFGHJKLZXCVBNM
2 'E' is not a possible letter for this spot, the
       valid letter is 'A'
3 'E' must be in the word
```

Listing 4. Various different retry prompts used in Wordle