Policy Gradients and Information-Theoretic Wordle Solving

Andrew Sung [*1] **Ethan Hersch** [*1] **Manav Kant** [*1]

Abstract

Wordle appears to be a simple game, but its underlying state space is large and decision-making requires balancing exploration and exploitation across a six-step horizon. While heuristic approaches based on information theory perform well, they do not adapt their strategy through learning. Reinforcement learning (RL), on the other hand, can optimize multi-step decision-making but often struggles with sparse rewards and large action spaces.

We combine information-theoretic reasoning with reinforcement learning to build an improved Wordle-solving agent. Prior work has demonstrated strong performance using greedy entropy maximization to select guesses that maximize expected information gain (Sanderson, 2022). RL-based approaches using actor-critic methods such as Proximal Policy Optimization (PPO) (Schulman et al., 2017) have been applied to learn Wordle strategies through self-play (Kho, 2022). We integrate these strategies, combining entropy-based priors into a PPO policy, allowing the agent to start from a strong heuristic baseline while learning corrections through policy optimization over the 6-step horizon.

We compare our approach against greedy entropy solvers and alternative RL baselines to see how much RL and entropy influence each other in playing this game. Our entropy-guided PPO agent achieves an average solution length of 3.51 guesses while maintaining a 100% win rate, whereas pure greedy entropy achieves an average solution length of 3.61 guesses (a small but consistent gain). Each of these is an improvement over pure RL methods (DQN and PPO) not using entropy. We further analyze the impact of reward shaping and dense reward signals on training stability and sample efficiency. Our results demonstrate that combining information-theoretic heuristics with policy-gradient RL provides a practical way to improve performance in structured reasoning games such as Wordle.

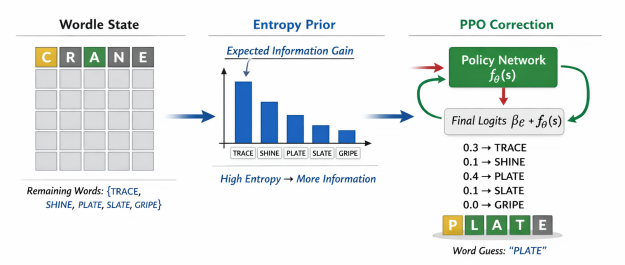

Figure 1. Overview of the entropy-guided PPO approach. Expected information gain from each guess is incorporated into the policy to guide action selection.

1. Introduction

1.1. Problem Setup

Wordle is a word-guessing game where a player has six attempts to identify a hidden five-letter word, receiving green/yellow/gray feedback after each guess. Despite its simplicity, it presents a challenging RL problem (Kraemer, 2022): 2,315 possible actions, a 6-step horizon, and extremely sparse rewards. Its state space is large, on the order of $2^{26}3^{5^{26}}$ (Kho, 2022). A random agent wins $\sim 0.3\%$ of games on a small test set; vanilla PPO with reward shaping trained for 50,000 episodes wins 99% of the time, taking 6 guesses on average to guess the correct word. This may sound promising; however, a Wordle solver should be achieving a perfect win rate, taking below 4 turns on average to guess correctly. As Sanderson says in his Wordle demo, "in Wordle, 4 is par" (Sanderson, 2022). One of our key evaluation metrics throughout this project will be in how few turns (on average) our agent can win Wordle on a test set of 500 games.

1.2. LLM Motivation

The success of large language models (LLMs) across many natural language tasks raises an important question: how well do they perform in structured decision-making environments? While LLMs demonstrate strong capabilities in reasoning and text generation, they are not always well-suited for tasks requiring systematic exploration of a combinatorial state space.

Wordle provides an illustrative example. Prior work experimenting with prompting-based approaches found that LLMs perform surprisingly poorly at the game (Gupta, 2023). We observe similar behavior when prompting GPT-5.2-Thinking to play Wordle. Figure 12 shows an example interaction (the hidden word is ATLAS). The model struggles to incorporate feedback from previous guesses efficiently and requires several minutes to produce each guess.

When playing 500 games against Gemini 2.5 Flash (utilizing the Vertex API), the agent only achieved a 3% win rate. LLMs also exhibit considerable latency while playing Wordle.

1.3. Buildup to Entropy-Guided PPO

Although the Wordle state space is large, it is structured and well-defined, making it amenable to reinforcement learning approaches. A natural first idea uses Q-learning with function approximation. However, this approach quickly encounters the deadly triad as we have function approximation, bootstrapping, and off-policy learning (Sutton & Barto, 2018). The difficulty is amplified in Wordle due to the large discrete action space and sparse rewards, which make accurate value estimation challenging. Increasing the width and capacity of the MLP does not fix this issue, and in practice the DQN agent fails to learn an effective strategy.

Prior work instead suggests that policy-gradient methods such as Proximal Policy Optimization (PPO) are better suited to this setting (Kho, 2022; Schulman et al., 2017). Following this direction, we train PPO-based agents to play Wordle, first starting with vanilla PPO, which does not leverage information theory. The vanilla PPO baseline also fails to learn a successful strategy.

Inspired by Arumugam & Griffiths (2025), which leverages information theory and LLMs for efficient Wordle guessing, we introduce an *entropy-guided* PPO agent that uses Shannon entropy to measure how much new information each guess adds, combining exact entropy scores with a learned correction term in the policy logits [Figure 1].

Our key insight is that information theory provides a natural prior: at each step, the optimal greedy strategy selects the word maximizing expected information gain (Sanderson, 2022; Shannon, 1948). We use this prior to initialize our RL policy, allowing it to start near-optimal and learn when to deviate for multi-step planning benefits. Our method converts a greedy entropy heuristic into a learnable policy prior that PPO can override, yielding a small but consistent improvement.

2. Related Work

Information-theoretic Wordle solving. Sanderson (2022) introduced a greedy entropy-maximization approach to Wordle. For each candidate guess, the solver computes the expected information gain:

$$\mathbb{E}[\text{Info}] = \sum_x p(x) \cdot \log_2\left(\frac{1}{p(x)}\right) = H(X)$$

where $p(x)$ is the probability of observing color pattern x given the current set of remaining words. The solver greedily picks the word maximizing this quantity at each step, achieving $\sim 3.6 - 3.7$ average guesses depending on word frequency weighting.

LLM-based exploration. Arumugam & Griffiths (2025) apply the Posterior Sampling Reinforcement Learning (PSRL) algorithm with large language models to sample from a posterior distribution and derive optimal exploration behavior for Wordle.

Deep RL for Wordle. Kho (2022) applies actor-critic methods to Wordle, demonstrating similar challenges with the large discrete action space without domain-specific guidance. Anderson & Meyer (2022) explore additional RL formulations for the problem.

Information-Seeking Rewards for LLM Agents. Recent work has explored augmenting reinforcement learning objectives for language agents with auxiliary rewards that encourage information-seeking behavior. Wan et al. (2025) introduce a curiosity-driven intrinsic reward in multi-turn RLHF that incentivizes a dialogue agent to actively infer user traits by improving the accuracy of a learned user model. This approach can be viewed as a form of reward shaping that encourages exploration and information acquisition during interaction. Their work focuses on personalization in conversational systems, but our method incorporates an entropy-based prior to encourage informative guesses in the Wordle setting during PPO optimization.

Prior approaches study Wordle by leveraging LLMs, RL, or information theory. They either rely purely on heuristics (greedy entropy) or purely on RL. Our approach combines both to show how much can be gained by using PPO for sufficient exploration over pure entropy.

3. Approach

3.1. Baselines

We first establish some simple baselines to compare our entropy PPO approach against.

A **random** agent selects uniformly from 2,315 words (≈0% win rate). An **LLM** (Gemini 2.5 Flash) wins 3% of the time. **DQN** uses a 3-layer MLP (1,024 layer width) with ϵ-greedy

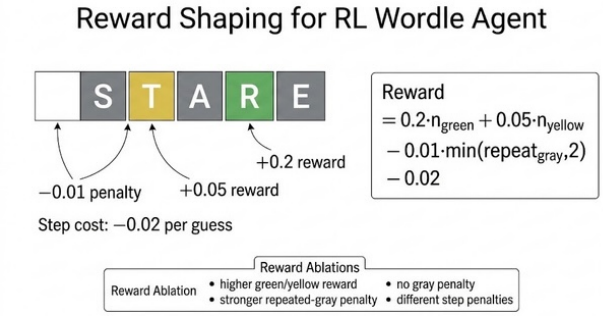

Figure 2. Our dense reward function. We penalize for each guess and reward for green and yellow characters. These weights can be adjusted and we also reward for a complete guess with +10.

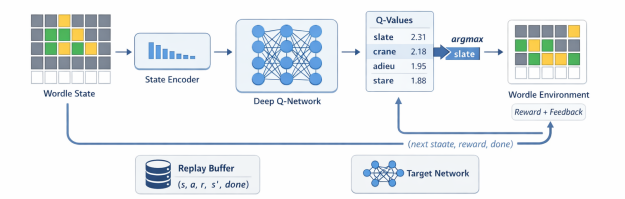

Figure 3. DQN-based Wordle approach

exploration but fails to learn meaningful Q-values. **Vanilla PPO** (Schulman et al., 2017) uses letter-aware dot-product attention over per-word embeddings ($5 \times 26 = 130$ features) with a 292-dim state encoding (green/yellow masks, eliminated letters, round). Reward is $r = -1$ per guess, $+10$ for winning. Despite this architecture, it achieves 0% win rate—the policy collapses with no positive reward signal to learn from. The reward function from the Wordle Gym (Kraemer, 2022) does not reward on the character level so we implement a new one.

3.2. Reward Shaping

To improve learning in the sparse-reward Wordle environment, we design a shaped reward function that provides both terminal and intermediate feedback. The original reward function in the Wordle Gym simply penalizes -1 for each guess and rewards $+10$ for a correct guess (Kraemer, 2022). This sparse reward signal makes it difficult for PPO and DQN to formulate a solving strategy as it does not receive feedback for how to value yellow and green characters. A better reward function should score on the character level.

We create our own reward function: each guess receives positive reward for correctly identifying letters and a small step penalty to encourage faster solutions [Figure 2]. Letters placed in the correct position (green tiles) receive a larger reward, while letters that are present in the word but in the wrong position (yellow tiles) receive a smaller reward. A constant negative step penalty is applied at every turn to discourage unnecessary guesses.

The reward at step t is described in Figure 2, with a considerable reward of +10 for solving the word. In this method, n_{green} and n_{yellow} denote the number of green and yellow tiles returned by the Wordle feedback for the current guess. By default, we also penalize each guess slightly (to incentivize solving in few steps). We also penalize repeated gray characters as this provides no added information for subsequent steps.

Reward shaping provides a denser learning signal than purely terminal rewards while maintaining the objective of solving the puzzle efficiently.

3.3. DQN & Entropy-Guided DQN

Our initial DQN for Wordle implementation first follows a Pytorch DQN tutorial (PyTorch Contributors, 2024) to understand basics.

As a value-based reinforcement learning baseline, we implement a Deep Q-Network (DQN) agent adapted to the structure of Wordle. Standard DQN estimates an action-value function

$$Q_\theta(s, a)$$

which predicts the expected discounted return obtained by taking action a in state s and following the learned policy thereafter [Figure 3]. Our Q network is sufficiently large (1,024 wide and 3 layers deep) but still struggles to learn an effective strategy. This agent uses our shaped reward described in Section 3.2. Wordle presents several challenges for vanilla DQN: the action space is large (thousands of possible guesses), rewards are sparse, and effective play requires information-gathering early in the episode.

To address these challenges, we combine standard Q-learning with an information-theoretic exploration prior. This allows the agent to leverage principled Wordle heuristics while still learning a value function through reinforcement learning.

State Representation. Rather than using raw board tokens, we construct a structured feature representation that captures the logical constraints revealed by the game:

- confirmed green letters by position
- yellow letters with their disallowed positions
- globally eliminated letters
- a one-hot encoding of the current round

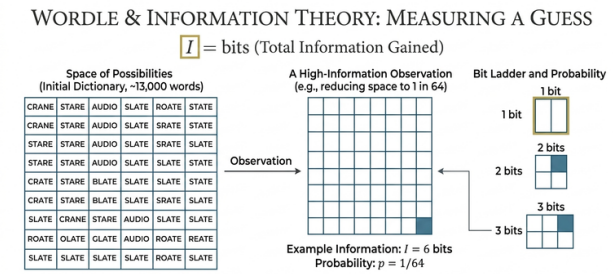

Figure 4. Explanation of precomputed entropy matrix. Across the entire Wordle vocabulary, we compute the entropy $\sum p(x) \log(\frac{1}{p(x)})$ to find total bits of information gained from each guess. This value measures how effectively a guess partitions the remaining solution space.

This representation produces a 294-dimensional feature vector. By explicitly encoding Wordle constraints, the network receives a compact symbolic description of the game state rather than needing to infer structure from raw observations.

Q-Network Architecture. The Q-function is parameterized by a multilayer perceptron with two hidden layers of size 256 and ReLU activations. The network outputs one Q-value per possible solution word. Restricting the action space to the official Wordle solution list reduces the output dimension and simplifies learning while preserving the sequential reasoning difficulty of the game. Note this network does not have to be as wide as the vanilla DQN architecture.

Entropy-Guided Action Selection. A key modification to standard DQN is the use of entropy-based exploration. Using a precomputed Wordle pattern matrix [Figure 4], we maintain the current candidate solution set and compute expected information gain for a subset of actions.

During exploration, instead of sampling uniformly at random, the agent samples guesses proportional to their expected entropy. During greedy selection, the action is chosen according to

$$a_t = \arg\max_a \left[Q_\theta(s_t, a) + \beta \cdot H(a) \right],$$

where $H(a)$ is the expected information gain of guess a and β controls the strength of the entropy prior. This encourages guesses that reduce uncertainty early in the game while still allowing the network to learn long-horizon value estimates.

Learning Algorithm. Training follows the standard DQN framework with a replay buffer and both target and online networks. We use the standard procedure to train a value-based reinforcement learning agent tailored to the structure of Wordle.

3.4. Vanilla PPO

We apply Proximal Policy Optimization (PPO) (Schulman et al., 2017) as our primary policy-gradient baseline. The agent uses a letter-aware state encoding of 292 dimensions, consisting of binary masks for confirmed green letters by position, yellow letters with their disallowed positions, globally eliminated letters, and a one-hot encoding of the current round. To handle the 2,315-word action space, we use dot-product attention over per-word embeddings derived from $5 \times 26 = 130$ letter features, producing a score for each candidate word.

With the default Wordle Gym reward (-1 per guess, $+10$ for winning), vanilla PPO achieves a 0% win rate. The sparse reward signal provides no gradient information about partial progress, causing the policy to collapse to a degenerate fixed point. To address this, we apply the dense reward function described in Section 3.2, which scores on the character level. With reward shaping, PPO achieves a 99.6% win rate but averages 6.00 guesses per game—barely solving within the allowed horizon. The agent learns that it should guess valid words but not how to strategically reduce uncertainty across guesses, motivating the entropy-guided approach.

3.5. Entropy-Guided PPO

Our main contribution incorporates information-theoretic reasoning directly into the PPO policy parameterization rather than relying on the agent to discover informative guessing strategies from scratch. The key idea is to use Shannon entropy as an action prior: at each game state, the expected information gain $H(X) = -\sum_x p(x) \log_2 p(x)$ (Shannon, 1948) of the feedback pattern distribution for each candidate guess measures how effectively that guess partitions the remaining solution space. Rather than using this signal as a reward bonus, we inject it directly into the policy logits.

Policy Formulation. Let $\mathbf{e}(s)$ denote the vector of expected information gains for all candidate guesses in state s, computed via a precomputed $2{,}315 \times 2{,}315$ pattern matrix, and let $f_\theta(s)$ denote the logits produced by the learned policy network. The final policy logits are

$$\ell(s) = \beta \cdot \mathbf{e}(s) + f_\theta(s),$$

where β is a learnable scalar (initialized to 10.0) controlling the strength of the entropy prior. The policy distribution over guesses is obtained via a softmax over $\ell(s)$ and optimized using PPO. This formulation allows the agent to start near the greedy entropy-maximizing strategy while learning corrections through f_θ that account for multi-step planning. Additionally, an exploitation bonus $\mathbf{1}[i \in \mathcal{R}]/|\mathcal{R}|$ biases the policy toward remaining viable words when few candidates remain, encouraging the agent to guess the solution directly

rather than continuing to gather information.

State Representation. The state encoding is extended to 301 dimensions: the base 292 features from vanilla PPO plus 9 information-theoretic features, including the fraction of remaining candidate words, the normalized entropy of the candidate distribution, per-round information gains from previous guesses, and cumulative information acquired so far. These features provide the policy network with a compact summary of the agent's current uncertainty.

Reward Function. Because Wordle provides sparse terminal feedback, we design a shaped reward that provides intermediate learning signals. Each guess receives a negative step penalty and a reward proportional to the information gained, measured as the reduction in the candidate solution set. The reward at step t is

$$r_t = -c + R_{\text{win}} \mathbf{1}_{\text{solved}} + \lambda \log_2 \left(\frac{n_{\text{before}}}{n_{\text{after}}} \right),$$

where $c = 1.0$ is the step penalty, $R_{\text{win}} = 10.0$ is the terminal reward, and n_{before}, n_{after} denote the candidate set size before and after the guess. The logarithmic ratio corresponds to the information gain, encouraging informative guesses early in the game while still prioritizing efficient solving.

Importantly, the entropy prior in the policy logits is distinct from this reward function. PPO's standard entropy regularization term is also retained during optimization to encourage policy exploration, but this term is unrelated to the Wordle-specific information gain used in the action prior.

4. Results

We evaluate all agents on a fixed test set of 500 Wordle games, reporting win rate (percentage of games solved within six guesses) and average number of guesses conditioned on successful games. Entropy PPO achieves 100% win rate and 3.51 avg rounds, outperforming the greedy entropy baseline (3.61) consistently.

A comparison of all results is displayed in Table 1

4.1. Overall Performance

Table 3 summarizes performance across all methods. Random and LLM-based agents perform poorly, achieving win rates of 0.3% and 3%. Vanilla PPO fails to learn a meaningful strategy, achieving a 0% win rate due to collapse under sparse rewards. Introducing reward shaping significantly improves PPO, achieving a 99.6% win rate, but requiring an average of 6 guesses, indicating inefficient play.

In contrast, information-theoretic methods perform substantially better. The greedy entropy baseline achieves a perfect 100% win rate with an average of 3.61 guesses. Our entropy-guided PPO agent consistently improves upon this baseline, achieving a 100% win rate with an average of 3.51 guesses.

Although the improvement over greedy entropy is modest in absolute terms, it is consistent across evaluation runs and demonstrates that reinforcement learning can refine an already strong heuristic policy.

Table 1. Performance comparison across agents (500 games). We observe win rate (% of games won) and the average number of guesses it takes to win (only wins). Notice vanilla PPO has no average rounds field because it doesn't win any games.

AGENT	WIN RATE	AVG ROUNDS
RANDOM	0.2%	6.00
GEMINI 2.5 FLASH	3%	6.00
VAN. PPO	0.0%	N/A
VAN. PPO (REWARD SHAPING)	99.6%	6.00
ENTROPY-GUIDED DQN	21.4%	4.65
GREEDY ENTROPY	100.0%	3.61
ENTROPY PPO (OURS)	**100.0%**	**3.51**

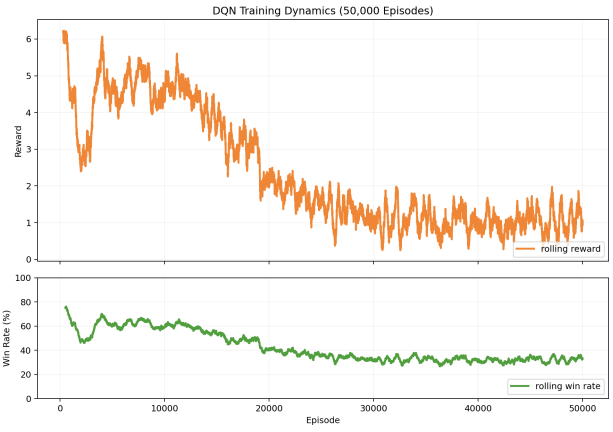

Figure 5. DQN training dynamics over 50,000 episodes. While early performance is relatively strong due to high exploration, both reward and win rate degrade as training progresses and ϵ decays, ultimately converging to a suboptimal policy. This highlights the difficulty of learning effective strategies in Wordle using value-based methods with standard reward shaping.

4.2. DQN Performance

The DQN agent performs poorly in this environment, achieving only a 21.4% win rate with an average of 4.65 guesses on successful games. As shown in Figure 5, both reward and win rate are initially high but deteriorate over time as exploration decreases, indicating that the agent converges to a poor local optimum. This explicitly shows how DQN

can fail to represent the space properly, converge to a poor solution with low reward, and attain a low win rate.

Episode	ϵ	β
50	0.995	4.979
6300	0.480	4.790
30000	0.071	0.714

Table 2. Training dynamics of the entropy-augmented DQN agent. The exploration parameter ϵ decays steadily over training, while the entropy coefficient β remains stable early on but decreases significantly in later training.

As shown in Table 4.2, the exploration parameter (ϵ) decays over training, consistent with what we would expect as an agent should explore less as more episodes run. The entropy coefficient (β) stays stable during the early stages of training and drops later, indicating reduced reliance on entropy-driven action selection and more dependence on the learned policy. The agent still fails to learn a strong policy.

This suggests neither exploration nor entropy-guided action selection is the primary bottleneck. The instability could stem from the deadly triad and be exacerbated by the large discrete action space and sparse rewards in Wordle (Sutton & Barto, 2018).

4.3. Effect of Reward Shaping

Reward shaping plays a critical role in enabling learning. Without shaping, PPO receives only terminal rewards and fails to discover successful trajectories, resulting in a degenerate policy with a 0% win rate. Introducing dense rewards based on green and yellow feedback provides a meaningful learning signal, allowing PPO to achieve near-perfect win rates [Table 1].

We evaluate several reward shaping variants to understand their impact on learning dynamics and final performance. Figures 6 and 7 show reward convergence and win rate over training, respectively.

Increasing the reward for correctly placed letters (green tiles) leads to the fastest convergence and highest final reward, indicating that stronger positive signals for informative guesses significantly improve learning. In contrast, variants that penalize repeated gray letters or increase the step penalty result in slower convergence and lower asymptotic performance. While these penalties are intended to discourage uninformative guesses, they appear to hinder exploration early in training.

Across all variants, we observe that reward shaping strongly influences both training speed and final policy quality. Notably, the baseline and green-weighted rewards achieve the

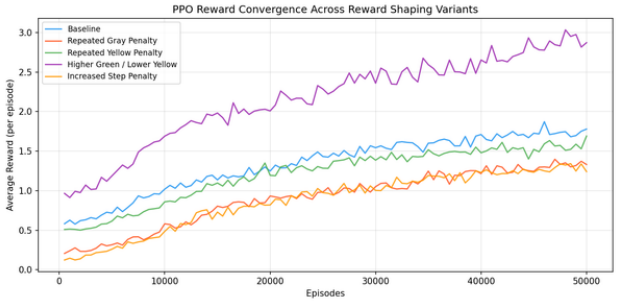

Figure 6. PPO reward convergence across different reward shaping variants. Increasing the reward for correctly placed letters (green) leads to faster and higher convergence, while penalizing repeated gray letters and increasing step penalties result in slower learning and lower asymptotic performance.

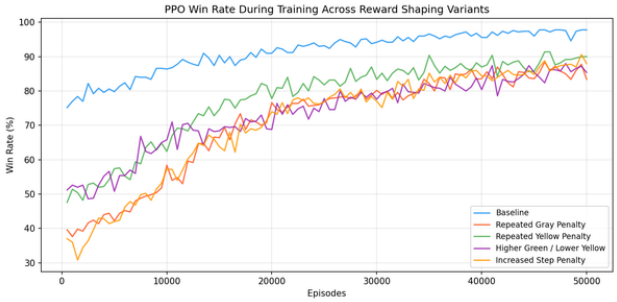

Figure 7. Win rate with PPO training across reward shaping variants. The baseline and green-weighted rewards achieve the highest final performance, while stronger penalties (repeated gray or increased step penalty) slow convergence and reduce final win rates.

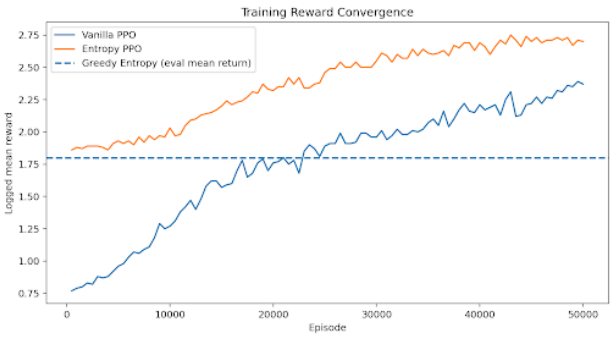

Figure 8. Training reward convergence for vanilla PPO and entropy-guided PPO over 50k episodes. Entropy-guided PPO achieves faster learning and higher final reward, indicating improved exploration and more effective policy optimization compared to vanilla PPO.

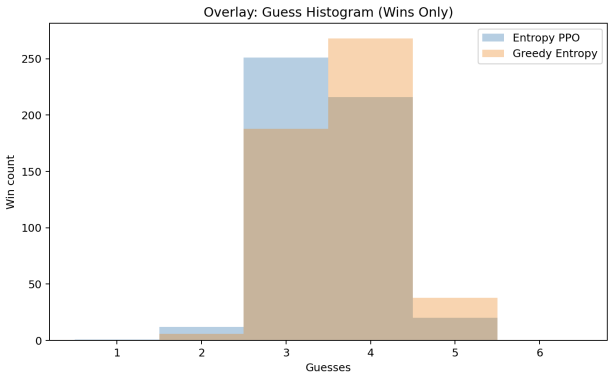

Figure 9. Distribution of number of guesses (wins only) for entropy-guided PPO vs greedy entropy. Entropy-guided PPO achieves a higher frequency of 3–4 guess solutions, while greedy entropy shows a heavier tail toward longer games, indicating less efficient solving.

highest win rates, suggesting that emphasizing positive feedback for correct information is more effective than imposing additional penalties.

These results highlight that while reward shaping is essential for enabling PPO to learn in the Wordle environment, overly aggressive penalties can degrade performance by restricting exploration and slowing policy improvement.

4.4. Entropy-Guided PPO

Our entropy-guided PPO agent combines the strengths of information-theoretic heuristics and policy optimization. By injecting expected information gain directly into the policy, the agent starts near a strong greedy baseline and learns corrections through training. In Figure 8, we can see entropy PPO starts around the greedy entropy baseline and continues to improve on top of the vanilla PPO method. Overall, it can solve Wordle in fewer guesses when evaluated over 500 games, achieving an average rate of 3.51 turns [Figure 9].

Our learned policy deviates from the greedy entropy strategy in approximately 16% of guesses. These deviations are not random; rather, they often correspond to situations where directly guessing a likely solution yields faster completion than further information gathering. By deviation, we mean the agent learns to override selecting the maximum entropy guess and relies on a learned PPO strategy.

Figure 11 illustrates this: while the greedy entropy solver continues to explore, the entropy-guided PPO agent exploits accumulated information to guess the correct word earlier, reducing the total number of guesses.

These results demonstrate that reinforcement learning primarily contributes by learning when to deviate from the entropy heuristic, rather than replacing it.

4.5. First Guess Analysis

Table 4 shows the top-ranked initial guesses under the entropy-guided PPO policy along with their expected entropy gain. The rankings closely align with those of the greedy entropy baseline (e.g., *raise*, *slate*, *crate*), indicating that the learned policy preserves the strong inductive bias of entropy maximization in early-game exploration.

This further supports the interpretation that reinforcement learning introduces targeted adjustments rather than fundamentally altering the strategy.

4.6. Summary

Overall, our results show that:

- Pure RL methods (DQN, PPO) struggle in the Wordle environment due to sparse rewards and large action spaces.
- Reward shaping improves learnability but does not yield efficient strategies.
- Greedy entropy provides a near-optimal baseline.
- Entropy-guided PPO achieves the best performance by combining a strong heuristic prior with learned corrections.

These findings suggest that incorporating domain-specific structure into RL policies can significantly improve performance in structured decision-making tasks.

Table 3. Performance comparison across agents (500 games). We observe win rate (% of games won) and the average number of guesses it takes to win (only wins). Notice vanilla PPO has no average rounds field because it doesn't win any games.

AGENT	WIN RATE	AVG ROUNDS
RANDOM	0.2%	6.00
GEMINI 2.5 FLASH	3%	6.00
VAN. PPO	0.0%	N/A
VAN. PPO (REWARD SHAPING)	99.6%	6.00
ENTROPY-GUIDED DQN	21.4%	4.65
GREEDY ENTROPY	100.0%	3.61
ENTROPY PPO (OURS)	**100.0%**	**3.51**

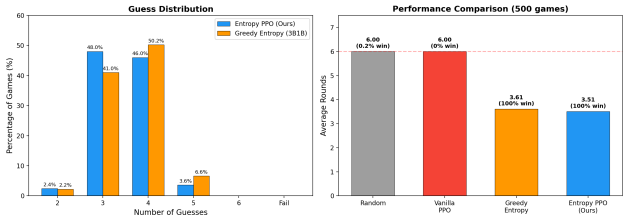

Figure 10. Left: distribution of guess counts for entropy-guided PPO and greedy entropy. Entropy-guided PPO shifts probability mass toward shorter solutions (3–4 guesses) and reduces longer games. Right: average number of guesses across agents, where entropy-guided PPO achieves the lowest average, outperforming both greedy entropy and vanilla PPO, indicating more efficient policies.

5. Analysis

In this work we study the challenge of applying reinforcement learning to Wordle, a deceptively simple game with a large discrete action space, sparse rewards, and a short but combinatorially complex decision horizon. We found that naively applying standard deep RL methods performs poorly in this setting. Both DQN and vanilla PPO struggled to discover effective strategies, largely due to the difficulty of exploring the large action space and the lack of informative reward signals early in training.

Our key insight is that information theory provides a powerful inductive bias for Wordle solving; adding RL on top of a greedy information-theoretic approach allows the agent to learn small policy corrections that improve long-horizon decision making. By incorporating an entropy-based prior directly into the policy logits, our entropy-guided PPO agent begins near a strong heuristic baseline while retaining the ability to learn policy corrections through reinforcement learning. Empirically, this hybrid approach outperforms both pure RL baselines and the greedy entropy solver. Across 500 evaluation games, our method achieves a 100% win rate with an average solution length of 3.51 guesses, consistently improving upon the greedy entropy baseline of 3.61 guesses (Sanderson, 2022). Analysis of policy behavior shows that the agent follows the entropy heuristic most of the time but deviates in approximately 16% of guesses, where learned corrections enable faster solutions by directly exploiting newly discovered constraints.

6. Conclusion

Despite these results, several limitations remain. First, our approach relies on a precomputed Wordle pattern matrix and exact entropy calculations, which assume a fixed vocabulary and full knowledge of the game mechanics. This limits the ability of the system to generalize to more open-ended language settings (or a Wordle game with a new vocabulary). Second, the improvements over greedy entropy are modest, suggesting that Wordle may already be close to optimally solvable with simple information-theoretic heuristics.

A few extensions remain. One could leverage information-theoretic LLMs. Following Arumugam & Griffiths (2025), we can use an LLM agent to generate candidate guesses conditioned on board state, estimate expected information gain via implicit word knowledge, and perform posterior updating via PSRL. This trades exact entropy computation for the ability to generalize beyond a fixed vocabulary. This future work would be interesting because we saw naively using LLMs to play Wordle is insufficient, but perhaps this is a viable strategy.

Additionally, while we experiment with PPO, Monte-Carlo tree search (MCTS) could be a possible RL algorithm to incorporate information theory. MCTS works well for domains with large search spaces and high branching factors (Silver et al., 2016). Wordle has this with its large state space and many possible letter changes at each position for each subsequent guess. Wordle also has a 6-step horizon, and MCTS can work well with shorter horizons since this limits the exponential growth of the tree and allows for more sampling at each step. MCTS can have high compute constraints, which is one of the reasons we decided to focus on PPO. With more compute, we hypothesize MCTS with information theory could produce a strong Wordle solver.

This work demonstrates that RL can benefit from incorporating domain-specific structure. For Wordle, combining information-theoretic priors with policy gradients yields a solver that slightly improves upon the best heuristic strategies while remaining trainable within an RL framework.

References

Anderson, B. J. and Meyer, J. Solving wordle with reinforcement learning. https://arxiv.org/pdf/

2202.00557, 2022.

Arumugam, D. and Griffiths, T. Toward efficient exploration by large language model agents. https://arxiv.org/pdf/2504.20997, 2025.

Gupta, M. Using large language models (llms) to play word games: Wordle. https://medium.com/@manavg/using-large-language-models-llms-to-play 2023. Medium blog post, accessed 2026-03-13.

Kho, A. Wordle solver: Actor-critic deep RL. https://andrewkho.github.io/wordle-solver/, 2022.

Kraemer, D. gym-wordle: An OpenAI gym compatible environment for wordle. https://pypi.org/project/gym-wordle/, 2022.

PyTorch Contributors. Reinforcement learning (dqn) tutorial. https://docs.pytorch.org/tutorials/intermediate/reinforcement_q_learning.html, 2024. Accessed: 2026-03-13.

Sanderson, G. Solving wordle using information theory. https://www.youtube.com/watch?v=v68zYyaEmEA, 2022. 3Blue1Brown.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Shannon, C. E. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 1948. https://people.math.harvard.edu/~ctm/home/text/others/shannon/entropy/entropy.pdf.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2 edition, 2018.

Wan, Y., Wu, J., Abdulhai, M., Shani, L., and Jaques, N. Enhancing personalized multi-turn dialogue with curiosity reward, 2025. URL https://arxiv.org/abs/2504.03206.

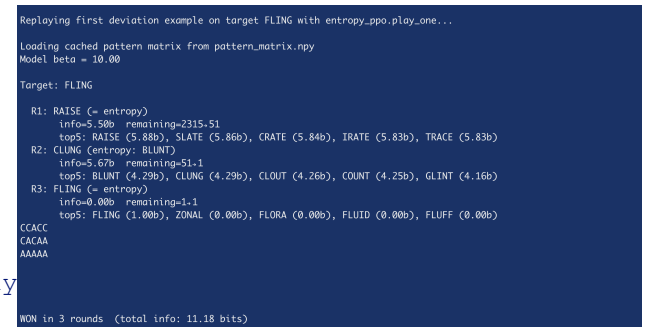

Figure 11. Deviation gameplay showing how entropy-guided PPO prioritizes information gain over immediate correctness. The agent selects high-entropy guesses to efficiently shrink the hypothesis space, enabling fast convergence to the solution.

7. Contributions

Ethan implemented DQN, made alterations to the reward function in the Gym environment, and ran experiments for PPO including first guess analysis and deviation analysis. He also contributed in writing all sections of the paper and the poster. Manav implemented the random agent, helped debug the PyTorch DQN tutorial for compatibility with the Wordle gym environment, ran experiments for PPO, and developed a theoretical understanding for the baselines' poor performance (i.e., large state/observation spaces, deadly triad). He contributed to writing the Approach and Results sections of the paper. Andrew implemented PPO, the greedy entropy agent, and the entropy-guided PPO agent, ran the reward shaping ablation experiments, trained/evaluated vanilla PPO baselines. He also helped with making the poster, helped with debugging the Wordle gym environment, and helped with writing the approach section of the paper.

Since we met frequently and a lot of the things were done together in person, equal contributions were made from all team members.

AI Usage: We used tools such as Claude Code and Codex to assist with generating boilerplate code for our methods, including entropy-guided PPO. These tools were also used to help implement specific experiments and generate plots. AI assistance was limited in the writing of the paper, primarily supporting minor editing of figure captions, improving clarity, and scaffolding portions of the approach section.

8. Appendix

Link to repo: https://github.com/ehersch/Wordle-Agent.

Wordle Gym: https://pypi.org/project/gym-wordle/.

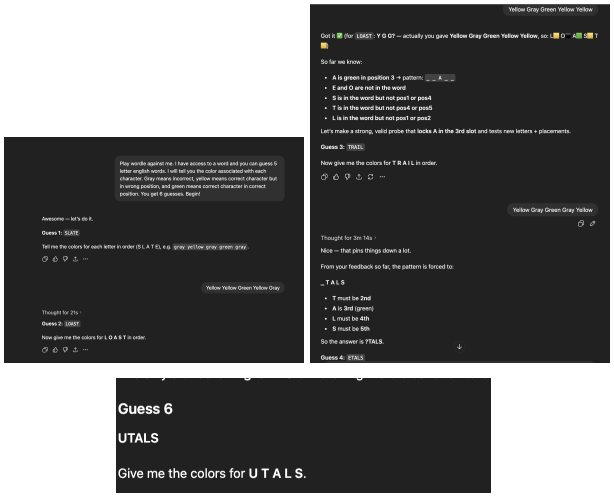

Figure 12. An example of GPT-5.2-Thinking playing Wordle. The model demonstrates strong reasoning ability but lacks consistency in optimizing for minimal guesses, motivating the need for reinforcement learning-based approaches.

Figure 13. Distribution of shaped rewards obtained by a random agent. The rewards are centered around a low mean (red line), reflecting limited information gain and lack of structured guessing, and serving as a baseline for evaluating learned policies.

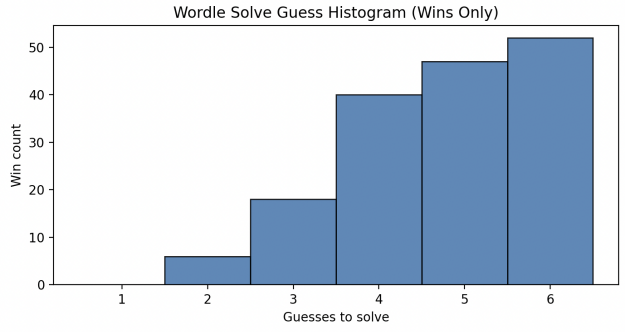

Figure 14. Distribution of guesses required by the DQN agent (wins only). The distribution is skewed toward higher guess counts, indicating inefficient policies and poor convergence compared to PPO-based methods.

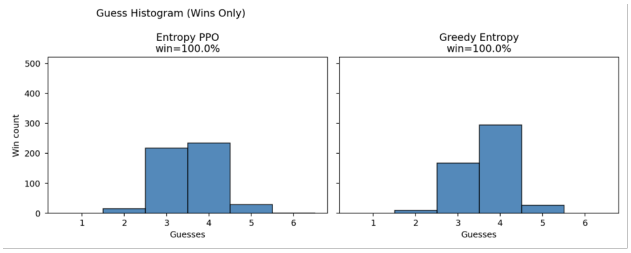

Figure 15. Win-only guess distributions for entropy-guided PPO and greedy entropy. Entropy-guided PPO produces a tighter distribution centered on fewer guesses, while greedy entropy exhibits greater variance and more frequent longer games.

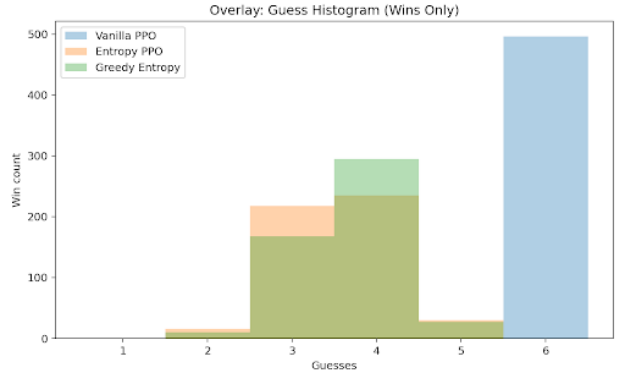

Figure 16. Overlay of guess distributions (wins only) for vanilla PPO, entropy-guided PPO, and greedy entropy. Vanilla PPO is heavily concentrated at 6 guesses, indicating near-failure to learn an efficient policy, while entropy-guided PPO and greedy entropy achieve substantially shorter solutions concentrated around 3–4 guesses.

Word	Score
raise	58.778
slate	58.554
crate	58.349
irate	58.318
trace	58.307
arise	58.209
stare	58.076
snare	57.702
arose	57.678
least	57.513
alert	57.456
crane	57.423
stale	57.390
saner	57.338
alter	57.136
later	57.067
react	56.961
leant	56.840
trade	56.819
learn	56.562
cater	56.474
roast	56.444
aisle	56.372
trice	56.345
scare	56.307

Table 4. Top 25 initial guesses ranked by expected information gain under the entropy-guided PPO policy. High-scoring words reflect balanced letter coverage and alignment with the learned policy's exploration strategy.

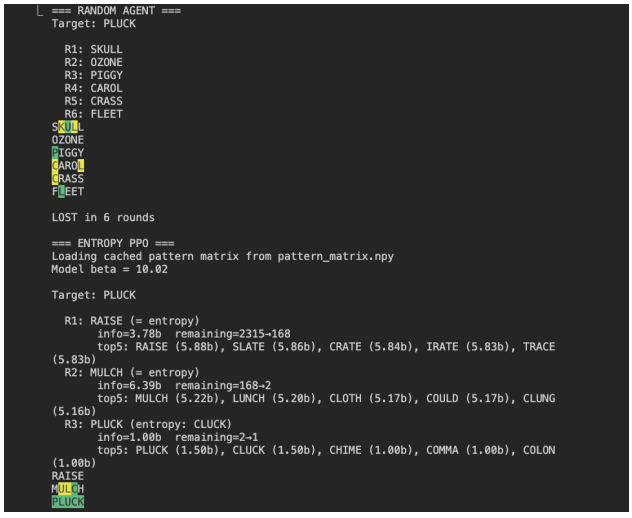

Figure 17. Gameplay on PLUCK. Random agent loses in 6 rounds; Entropy PPO wins in 3, deviating from greedy entropy (CLUCK) to guess the solution directly.