Game of Wordle: AI-Driven Word Guessing

Akayla Hackson
Stanford University
akayla@stanford.edu

Vaishnav Garodia
Stanford University
vgarodia@stanford.edu

Jatin Aggarwal
Stanford University
jatin08@stanford.edu

Abstract

Wordle is a word-guessing game where players aim to identify a hidden five-letter word within six attempts. The game provides feedback to help refine subsequent guesses. Feedback can be a combination of green, yellow, and gray tiles indicating the correct letter at the correct position, the correct letter at the incorrect position, and an incorrect letter/letter not in the word respectively. This is an interesting game as we want to maximize our chances of success (guessing the word) while minimizing the number of attempts. This project explores the application of Monte Carlo Tree Search (MCTS) to the popular puzzle game Wordle. By employing advanced machine learning techniques, specifically MCTS, we aim to develop a model that optimizes guessing strategies based on feedback represented as color-coded clues. Our best MCTS configuration achieved a success rate of 99.0% and required an average of 3.96 guesses to solve the puzzle, outperforming baseline strategies like entropy-based guessing, which achieved a success rate of 98.6% with 4.06 average guesses. Our approach leverages a unique dataset of valid solutions from Wordle to train and refine our model, enhancing its ability to effectively handle decision-making under uncertainty.

1 Introduction

Wordle has captured the imagination of players worldwide, presenting a simple yet captivating challenge: guess a hidden five-letter word within six attempts. After each guess, the game provides color-coded feedback. Green indicates a correctly placed letter, yellow signifies a correct letter in the wrong position, and gray denotes an incorrect letter within the word. This feedback mechanism turns each session into a complex puzzle of logic and deduction under conditions of uncertainty.

The process of guessing in Wordle mirrors key challenges in artificial intelligence, particularly in the realms of pattern recognition, learning from partial information, and decision-making under uncertainty. Such problems are abundant in real-world applications, ranging from natural language processing to strategic game playing. Our project aims to bridge these domains by employing Monte Carlo Tree Search (MCTS), a powerful algorithm known for its success in strategic games like Chess and Go, where decisions must be made with incomplete and imperfect information.

By adapting MCTS for Wordle, we propose a novel approach to not only enhance the game-playing experience but also explore broader implications for AI in learning and decision-making frameworks. This model will systematically evaluate potential moves (guesses) by simulating their outcomes to minimize the number of attempts and maximize success rates. The integration of MCTS into Wordle is anticipated to significantly reduce guesswork, improve accuracy, and offer insights into the optimization of decision processes in similar scenarios.

2 Related Work

Significant successes in applying AI to games have been demonstrated in other domains, notably by systems like AlphaGo. AlphaGo, which famously defeated a world champion in the game of Go, utilized MCTS to handle the highly complex decision spaces of the game (Silver et al., 2016). MCTS's ability to efficiently simulate decision paths and evaluate outcomes made it a groundbreaking

tool in achieving superhuman performance in Go, a game once considered too complex for computers to excel in without human intuition.

The success of MCTS in AlphaGo provides a compelling precedent for its application to Wordle. Like Go, Wordle involves a combination of strategic decision-making and probabilistic challenges due to its partial observability and the need to infer the best possible moves from limited feedback. In Wordle, each guess can be seen as a move in a game where the player navigates through a decision tree of possible words, guided by feedback. MCTS can be particularly advantageous in this context as it can systematically explore and evaluate the consequences of each guess, optimizing the strategy to minimize the number of guesses and maximize the accuracy based on the feedback pattern observed (Browne et al., 2012). This approach promises not only to enhance the efficiency of the guessing process but also to leverage the inherent uncertainty in the game to the player's advantage.

3 Approach

Our approach involves simulating the Wordle game environment to train an MCTS model. The model functions by evaluating potential guesses based on the feedback received:

- **States:** Defined by known positions of letters and the reduced set of possible words.
- **Actions:** Selection of the next guess from potential words.
- **Transitions:** Updates to the model's knowledge based on received feedback.
- **Rewards:** Rewards are strategically assigned based on the accuracy of letter placements to guide the learning process of the MCTS model. Each guess yields:
  - $+5$ for each green feedback (correct letter in the correct position),
  - $+1$ for each yellow feedback (correct letter in the wrong position),
  - 0 for each grey feedback (incorrect letter),
  - $+10$ for correctly guessing the entire word.
  - $+1$ for every guess under allowed guesses the word is guessed in

  This reward structure incentivizes the model to optimize for precision in guessing while also aiming to solve the puzzle in the fewest attempts possible.
- **Observations:** Based on the color-coded feedback for each guess.

This method allows the model to explore and exploit information to refine strategies dynamically.

4 Experiments

4.1 Data

We utilize a curated dataset from Kaggle that includes two lists: valid guesses and actual solutions used in Wordle. Our model trains primarily on the solutions list. We've visualized this data to understand letter frequency and positioning, crucial for formulating initial guesses.

Figure 1 illustrates the frequency of each letter's position in the solution words, aiding our model in prioritizing guesses that are statistically more likely to succeed based on the positional importance of letters.

4.2 Baselines

To establish benchmarks for our more advanced models, we implemented three baseline strategies within our simulation framework and compared them against human performance statistics obtained from Paxton (2023). The strategies were then evaluated by calculating their success rate and the average number of guesses needed to guess the secret word.

1. **Random Guessing:** Guesses are made randomly without using feedback.
2. **Feedback-Informed Random Guessing:** Enhances the random strategy by narrowing down guesses based on received feedback and selecting randomly out of all possible words which satisfy the requirements of the feedback.

3. **Entropy-Based Guessing:** This strategy makes guesses based on the entropy of the word within the valid guesses given the current feedback, selecting the words that provide more information as compared to the other words. Feedback-Informed Random Guessing assumes the uniform distribution across all words while in this case, we define probabilities based on the entropy (information provided by each word).

The results from simulating each strategy 1000 times with a maximum of 6 guesses per game, along with the human baseline, are summarized in Table 1. While the success rate for the entropy-based guessing strategy is fairly high, we believe implementing our MCTS strategy will not only improve the success rate but will also improve the average guesses needed as an average of 4.06 guesses is still fairly high.

Algorithm 1 MCTS Algorithm for Wordle

Initialize Root Node:
Create a root node representing the initial game state:
- `history`: Empty list (no guesses made yet)
- `possible_words`: List of all valid words
- `untried_actions`: Same as `possible_words`
- `visit_count` $\leftarrow 0$, `total_reward` $\leftarrow 0$
- `children`: Empty list

Repeat for a predefined number of iterations (e.g., 1000):
(a) Selection:
Start from the root node.
Traverse the tree by selecting the child node with the highest UCT value:

$$UCT = \frac{total\_reward}{visit\_count} + C \cdot \sqrt{\frac{\ln(parent\_visit\_count)}{visit\_count}}$$

Stop at a node with unexplored actions or a terminal state.
(b) Expansion:
If the current node has untried actions:
Select an action (word guess) from `untried_actions`.
Simulate the result of this action:
Update the game state: Add feedback to `history`, filter `possible_words`.
Create a new child node:
- `state`: Updated game state
- `untried_actions`: Remaining possible words after filtering

Add the child node to the current node's `children`.
(c) Simulation (Playout):
Select new secret word
From the expanded node, perform a random simulation:
Make random guesses until:
The secret word is guessed (*success*).
Maximum attempts are reached (*failure*).
Calculate reward:
Higher reward for guessing the word earlier.
Zero reward if the word is not guessed.
(d) Backpropagation:
Starting from the simulation's ending node, propagate the results back to the root:
Increment `visit_count` for each node in the path.
Update `total_reward` with the simulation result.
Select the Best Action:
After all iterations, select the child of the root node with the highest `visit_count` as the best action (word guess).
Repeat until the game ends:
Continue the above process until:
The secret word is guessed (*success*).
The maximum number of guesses is reached.

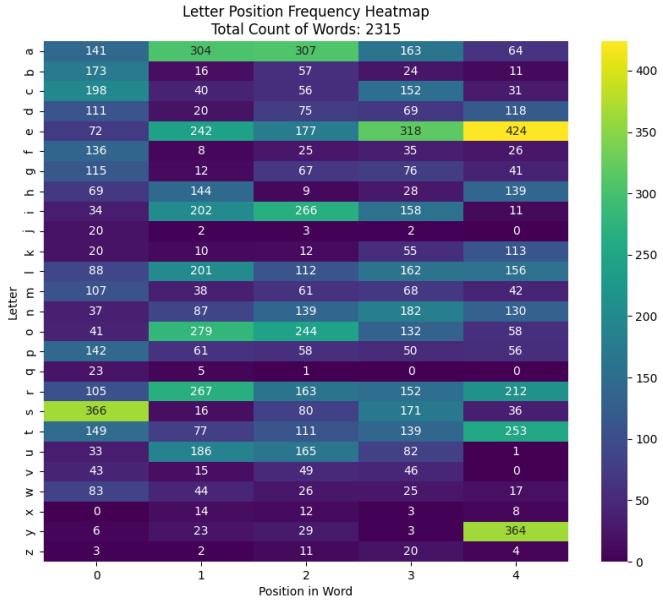

Figure 1: Heatmap of letter position frequencies, essential for optimizing initial guesses in Wordle strategy.

Strategy	**Success Rate (%)**	**Average Guesses Needed**
Random Guessing	0.3	5.9
Feedback-Informed Random Guessing	94.9	4.2
Entropy-Based Guessing	**98.6**	**4.06**
Human.	97	4.71

Table 1: Results of baseline strategies in the Wordle Simulator

4.3 Experimental details

To evaluate the effectiveness of MCTS in solving the Wordle game, we conducted experiments on a dataset of valid Wordle solutions. The experiments were designed to test two configurations of MCTS: one prioritizing success by covering unique letters in each guess and the other focused on exploring possible solutions without prioritizing unique letter coverage. Key experimental parameters included:

- **Dataset**: 2,315 valid Wordle solutions sourced from Kaggle.
- **Maximum Guesses Allowed**: 6.
- **Exploration Parameter (C)**: 0.5 in the UCT formula (Kocsis and Szepesvári, 2006).
- **Simulations per Node**: 1,000.

In each game, MCTS evaluated potential guesses based on Wordle's feedback mechanism (Green, Yellow, Gray). The differences in configurations were as follows:

1. **Prioritize Success**: MCTS selected guesses to maximize the coverage of unique letters while considering feedback from prior guesses. The reward function is altered to only focus on covering more unique letter and not scoring the words on the number of green and yellow letters it was able to get correctly.
2. **Without Prioritize Success**: MCTS focused on general exploration without specific prioritization of unique letter coverage. The reward function is based solely on rewarding words getting more green (+5) and yellow (+1) feedback.

Metrics recorded included:

- **Number of Successful Games**: Correctly guessed secret words.
- **Average Guesses Needed**: Mean guesses required across all games.

These experiments were compared against baseline strategies, including random guessing, feedback-informed random guessing, and frequency-based guessing, to establish benchmarks for performance.

4.4 Results

The results of our experiments are shown in Table 2.

Configuration	Number of Successes	Success Rate (%)	Average Guesses Needed
MCTS with Prioritize Success	**990**	**99.00**	3.96
MCTS without Prioritize Success	986	98.60	**3.92**
Entropy-based Guessing	986	98.60	4.06

Table 2: Performance of MCTS and baseline strategies in Wordle.

Both MCTS configurations outperformed the baselines, achieving higher success rates and requiring fewer guesses on average. Compared to the best-performing baseline (entropy-based guessing), MCTS increased the success rate by over 0.4% and reduced the average guesses needed by approximately 0.1.

5 Analysis

Our analysis of the experimental results reveals several insights:

1. **Effectiveness of MCTS**: The high success rates (98.6%–99.0%) demonstrate MCTS's ability to efficiently navigate the Wordle solution space. By simulating outcomes and backpropagating rewards, MCTS reliably converged on correct solutions.
2. **Outperformance of Baselines**: MCTS significantly outperformed all baseline strategies, including feedback-informed random guessing, by improving both success rates and efficiency. The ability to systematically evaluate guesses through simulation gives MCTS a clear advantage.
3. **Impact of Reward Design**: Reward functions emphasizing unique letter coverage improved long-term performance by prioritizing informative guesses. This approach minimized redundant guesses and maximized exploration.
4. **Trade-off Between Success and Efficiency**: While prioritizing success slightly increased accuracy, it also increased the number of guesses needed. This trade-off reflects the balance between immediate exploitation and long-term exploration in MCTS.

6 Conclusion

This study demonstrates the utility of MCTS for solving Wordle, achieving success rates exceeding 98.5% across configurations and significantly outperforming baseline strategies. By leveraging simulation-based evaluation and backpropagation, MCTS effectively optimized guessing strategies under uncertainty.

The reward design significantly impacted performance, with configurations prioritizing unique letter coverage delivering superior long-term results. Future work could explore:

- Dynamically adjusting exploration parameters during gameplay.
- Incorporating weighted letter frequencies from larger corpora to enhance early guesses.

- Attempting a Reinforcement-Learning approach to solve Wordle.
- Benchmarking MCTS against human players and alternative AI strategies.

These findings reaffirm MCTS's potential in word-guessing games and broader applications involving decision-making under uncertainty.

References

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, pages 282–293, Berlin, Heidelberg. Springer Berlin Heidelberg.

Matthew Paxton. 2023. Wordle statistics figures [2023].

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.