AI Wordle Solver

Artificial Intelligence - CSC361 Group Project

Abdullah AlHudaif 444102711
Muath AlMansoor 444103027
Omar AlKhamis 444102582

Supervised by: Dr. Hassan Mathkour

Contents

1 Overview

1.1 Problem Description

Wordle is a word-guessing game in which players deduce a hidden word based on feedback from previous guesses. An effective strategy requires selecting guesses that provide the most information, reducing the number of attempts needed.

1.2 AI Approaches

This project focuses on developing AI agents that efficiently solve Wordle puzzles. To address this problem, four AI-driven approaches are implemented:

- **Constraint Satisfaction Problem (CSP)** – Narrows down possible words by enforcing logical constraints based on previous guesses.
- **Letter Frequency Analysis** – Prioritizes words with high-frequency letters to maximize information gain.
- **Bayesian Updating** – Continuously updates probabilities of word candidates using Bayes's Theorem based on observed clues.
- **Entropy Maximization** – Selects guesses that maximize information gain to reduce the search space optimally.

1.3 Evaluation Criteria

Each AI agent is evaluated based on three key performance metrics:
solution time duration, win rate, and average number of guesses per win.
The objective is to determine which approach is most effective in solving Wordle with minimal attempts while maintaining efficiency.

2 Project Description

Wordle's simplicity and clear feedback make it a great choice for testing AI strategies. Unlike games like Tic-Tac-Toe, which have only a few distinct games after accounting for rotations and repetitions, Wordle presents a much larger space of possible puzzles.

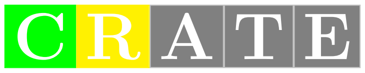
This diversity makes it a stronger benchmark for evaluating decision-making algorithms. The techniques explored here have broader applications in fields such as optimization problems, decision-making, and enhancing search efficiency in various types of AI agents.

2.1 How Wordle Works

In Wordle, the objective is to guess a 5-letter word within six attempts. [1] After each guess, feedback is given based on the accuracy of the guessed word. The feedback uses three colors:

- **Green**: The letter is correct and in the correct position.
- **Yellow**: The letter is correct but in the wrong position.
- **Grey**: The letter is not in the word at all.

Here's an example of a guess and its feedback, assuming the target word is "Curds":



In this example:

- The letter `C` is correct and in the right position (green).
- The letter `R` is in the word but in the wrong position (yellow).
- The letters `A`, `T`, and `E` do not appear in the word at all (grey).

2.2 Related Work

Several strategies have been developed to solve Wordle puzzles effectively. Some key ones include:

- **Deterministic Approach (Graph Formation):** Lahiri, Shah, Nandakumar, and Agarwal [2] explored Wordle strategies using graph-based models. Their approach involved:
  - Representing each word as a node in a graph.
  - Connecting words with edges if they share no letters or only a few.
  - Using Wordle feedback (green, yellow, gray) to gradually eliminate groups of incompatible words.
- **Best Starting Words:** Anderson and Meyer [3] looked at thousands of Wordle games to find strong starting words specifically for human players. They noticed:
  - Certain letters show up more often in specific spots—like 'S' at the beginning and 'E' at the end.
  - Good opening guesses include SAREE, SEINE, and POUPT.
  - Using this approach, human players could solve around 65% of games within six tries.
- **Reinforcement Learning:** Bhambri, Bhattacharjee, and Bertsekas [4] approached Wordle using reinforcement learning by:
  - Modeling Wordle as a decision problem under uncertainty.
  - Employing a rollout algorithm built on a base heuristic to simulate future outcomes.
  - Achieving near-optimal performance with relatively low computational cost.

2.3 Proposed Approaches

The system implements four distinct AI strategies for Wordle solving, all interacting with a standard game environment:

- **Bayesian Probability Agent**
  - Maintains a normalized probability distribution over all possible solutions.
  - Updates the probability of each candidate being the correct word using a likelihood heuristic based on how well the predicted feedback matches the actual feedback, applying an exponential decay function to score mismatches.
  - Gradually eliminates candidates with lower probabilities.
  - Chooses the next guess based on the highest posterior probability.
- **Entropy Agent**
  - Chooses guesses that maximize expected information gain.
  - Utilizes precomputed entropy values for initial moves to enhance efficiency.
  - Switches to real-time entropy calculations as the candidate pool decreases.
- **Constraint Satisfaction Agent**
  - Iteratively filters out impossible candidates based on exact feedback.
  - Randomly selects a guess from the remaining candidates.
- **Letter Frequency Heuristic Agent**
  - Prioritizes words that contain the most frequently occurring letters in the remaining candidates.

Note: The Bayesian, Frequency, and Entropy agents apply the same candidate filtering strategy as the Constraint Satisfaction Agent, but differ in their methods of selecting the next guess.

3 Discussion

The **Entropy Agent** achieved the best overall performance by effectively maximizing information gain, though it remained the slowest among all agents. A shared challenge across all agents was handling repeated letters, as in words like "books". Another common difficulty arose when only one guess remained while multiple candidates were still possible; requiring the agent to rely on its heuristic to select a guess, yet still leaving the outcome partially up to chance. Additionally, the **Entropy Agent** specifically faced the challenge of balancing pre-computation with real-time processing costs.

3.1 Comparison of Agents

The performance evaluation of the four agents—Entropy, Frequency, Bayesian,
and Constraint Satisfaction (CSP) revealed distinct differences in their efficiency, accuracy, and computational cost. Each agent was tested over 100 trials, with each trial consisting of 100 games, and the metrics observed include the average number of guesses per win, win rate, and average computation time per game.
The Entropy Agent demonstrated the highest overall performance in terms of accuracy and efficiency, achieving a perfect win rate of 100% and the lowest average number of guesses per win at 3.92 guesses (Figure 3.1). This outcome validates its core strategy of maximizing information gain. However, this advantage came at the expense of runtime performance, with an average computation time of 6.47 seconds per game (Figure 3.2), significantly higher than all other agents. This highlights a trade-off between decision quality and real-time applicability.

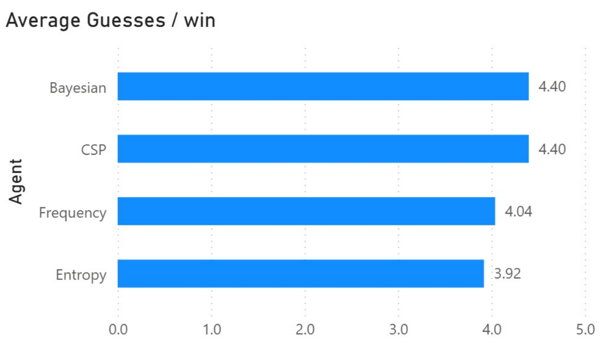


Figure 3.1: Average number of guesses per win for each agent, based on 100 trials of 100 games each.

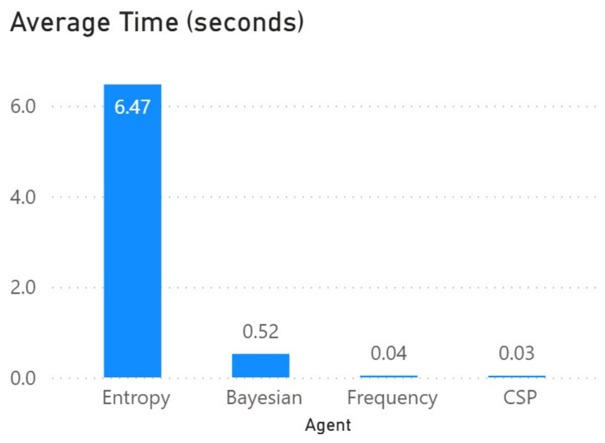


Figure 3.2: Average computation time per game (in seconds) for each agent, measured over 100 trials of 100 games each.

The **Frequency Heuristic Agent** offered a strong balance between performance and speed. With an average of 4.04 guesses per win and a win rate of 96%, it outperformed both the CSP and Bayesian agents in effectiveness, while maintaining a runtime of only 0.04 seconds per game. This makes it a practical alternative for applications requiring both efficiency and accuracy.

The **Bayesian Agent**, employs a probabilistic approach to refine its guesses, updating a probability distribution based on feedback. This method results in a lower win rate (94.81%) compared to other agents; due to the agent distributing probability across multiple candidates, which deflects focus from the most likely word and increases the average number of guesses. Additionally, the need to update and normalize probabilities for all candidates slightly increases computational cost.

The **CSP Agent**, the simplest of the four, excelled in computational speed, with an average time of just 0.03 seconds per game. It achieved a win rate of 94.99%, the second lowest, just above the Bayesian Agent (94.81%). Although this is a solid performance, the agent's non-deterministic guessing strategy—resulting from the randomness introduced after constraint filtering—led to a higher average number of guesses per win (4.40). This indicates that, while the CSP Agent is quick, its random decision-making process reduces its overall efficiency, making it less optimal compared to other agents in terms of guessing accuracy per win.

Agent	**Avg. Guesses / win**	**Win Rate**	**Avg. Time / game**
CSP	4.40	94.99%	**0.03s**
Frequency	4.04	95.63%	0.04s
Bayesian	4.40	94.81%	0.52s
Entropy	**3.92**	**100%**	6.47s

Table 1: Comparison of agents across average guesses per win, win rate, and time per game, based on a **2.3k-word dataset.**

3.2 Comparison of Agents and Human Players

Human players typically solve Wordle in an average of 4.1 attempts [5].

In contrast, AI agents have the advantage of access to the complete word list, which allows them to easily narrow down possible solutions.

For instance, the Entropy Agent averages 3.92 attempts per win, which is close to the human average of 4.1 guesses, but achieves a 100% success rate.

This comparison shows the clear advantage that AI agents have over human players when given access to a comprehensive word database.

3.3 Limitations and Fixes

A key limitation commonly observed in both the **Bayesian Agent**
and the **Frequency Heuristic Agent** was their handling of tie-breaking when multiple candidates had identical scores.
This occurred frequently in cases where several candidates shared the same maximum frequency or probability score.

Due to Python's `max()` function returning the first match in the list when scores are equal, both agents consistently selected the same incorrect word when multiple top candidates were present. As a result, certain target words were never correctly guessed. For example:

- **Frequency Heuristic Agent**: consistently failed to solve words like "stare", "pores", and "hutch".
- **Bayesian Agent**: consistently failed on words like "ratty" and "woody".

In all these cases, the correct word was not the first in the candidate list, and the agents defaulted to selecting the first option returned by `max()`, leading to 100% failure on those inputs.

To address this limitation, a tie-breaking fix was introduced: when multiple candidates shared the highest score, the agent would randomly select one among them rather than defaulting to the first. While this adjustment does not guarantee a correct guess, it mitigates the deterministic failure mode observed in similar candidate sets (e.g., "ratty" vs. "batty" or "pores" vs. "bores").

This change increases the probability of success from zero to a value proportional to the number of guesses remaining divided by the number of top-scoring candidates. For instance, with one guess left and four equally likely candidates, the success chance improves from 0% to 25%. Despite not being a perfect solution, it introduces variability and gives the agent a non-zero chance in otherwise unwinnable scenarios.

4 Conclusion

Among the four strategies, the **Entropy Agent** delivered the best performance, solving all games with the fewest average guesses. However, its considerably higher computational time introduces a trade-off. The **Frequency Heuristic Agent** emerged as a strong alternative, striking a good balance between accuracy and efficiency.

The **Bayesian Agent**, while employing a robust probabilistic approach, achieved a lower win rate and required more guesses due to its distributed probability method, and the **CSP Agent**, though faster, required more guesses per win due to its non-deterministic guessing strategy.

Future work could focus on enhancing the heuristics for all agents, specifically to improve their performance when handling words with repeated letters. This would help address challenges such as those seen in words such as “eerie”, or “alias”.

Additionally, further optimization of entropy calculations could improve the efficiency of the **Entropy Agent**, making it more practical for real-time applications.

Another area for improvement is the tie-breaking mechanism when multiple candidates share the highest score, as this could enhance the chances of success in such cases.

References

[1] Josh Wardle. *Wordle*. https://www.nytimes.com/games/wordle/index.html. Online game published by The New York Times. 2021.

[2] Aditya Lahiri et al. *Deterministic Algorithmic Approaches to Solve Generalised Wordle*. 2023. arXiv: 2305.14756 [cs.DS].

[3] Benton J. Anderson and Jesse G. Meyer. *Finding the optimal human strategy for Wordle using maximum correct letter probabilities and reinforcement learning*. 2022. arXiv: 2202.00557 [cs.CL].

[4] Siddhant Bhambri, Amrita Bhattacharjee, and Dimitri Bertsekas. *Reinforcement Learning Methods for Wordle: A POMDP/Adaptive Control Approach*. 2022. arXiv: 2211.10298 [cs.AI].

[5] Dean Talbot. *Wordle Scores by Country and User Statistics*. https://wordsrated.com/wordle-scores-by-country/. Published: October 17, 2023. Accessed: 2025-04-05. 2023.