

Wordle with Solver

Haotian Fang, Yu Ma, Yiyao Yin, Che Zhai, Yinuo Zhu

November 26, 2024

Contents

Code Accessibility

Overview

Background

Minimax-Based Player

Frequency-Based Player

Entropy-Based Player

Decision-Tree-Based Player

Result

References

Use of AI

Code Accessibility

All the code you can found on my GitHub @WncFht.
Here is the repository: Wordle-with-Solver
And we have already released version 1.0.0.

Overview

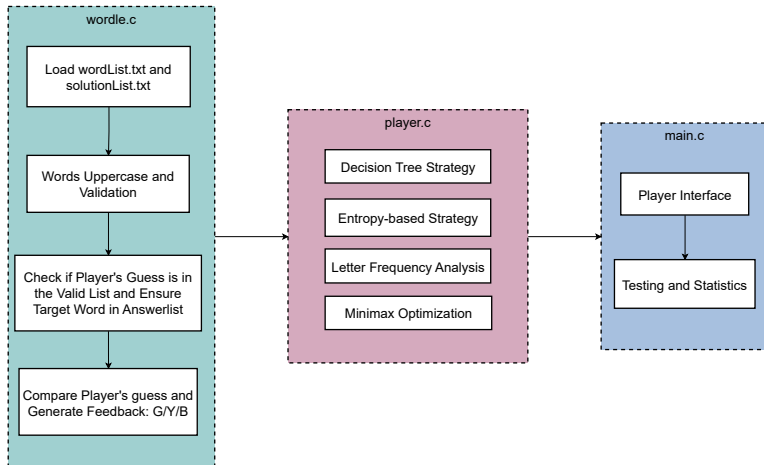


Figure: Overview

Background

1. Minimax

- ▶ Minimizes the maximum possible remaining solutions.
- ▶ Good for worst-case scenario optimization.

2. Frequency-based

- ▶ Uses letter frequency analysis.
- ▶ Considers both position-specific and overall frequencies.

3. Entropy-based

- ▶ Uses information theory to maximize information gain.
- ▶ Starts with "STARE" as the first guess.

Minimax-Based Player

$$N_i(w) = \sum_{s \in S} \delta(P(w, s), P_i) \quad (1)$$

$$M(w) = \max_i N_i(w) \quad (2)$$

```
1  for (int i = 0; i < wordCount; i++) {
2      generate_pattern_counts(wordList[i], pattern_counts);
3
4      int max_remaining = 0;
5      for (int j = 0; j < PATTERN_COUNT; j++) {
6          if (pattern_counts[j] > max_remaining) {
7              max_remaining = pattern_counts[j];
8          }
9      }
10
11     if (max_remaining < min_worst_case) {
12         min_worst_case = max_remaining;
13         strcpy(best_guess, wordList[i]);
14     }
15 }
```

Frequency-Based Player

$$\text{score}(w) = \sum_{j=1}^k (2 \cdot \text{letter_freq}[L_j][j]) + \sum_{L \in w \text{ (unique)}} \text{total_freq}[L] \quad (3)$$

$$w_{\text{best}} = \arg \min_w M(w) \quad (4)$$

```
1  for (int i = 0; i < wordCount; i++) {
2      float score = 0.0f;
3      int used[26] = {0};
4
5      for (int j = 0; j < WORD_LENGTH; j++) {
6          int letter = wordList[i][j] - 'A';
7          score += letter_freq[letter][j] * 2.0f; // Position-specific score
8
9          if (!used[letter]) {
10             score += total_freq[letter]; // Overall letter frequency score
11             used[letter] = 1;
12         }
13     }
```

Understanding Information Entropy

1. Information Entropy Calculation Formula

$$H(X) = - \sum_{i=1}^n p(x_i) \log_2 p(x_i) \quad (5)$$

where $p(x_i)$ is the probability of event x_i occurring.

Entropy Strategy Overview

Initialize solution set. For each guess:

- ▶ Update possible words.
- ▶ Calculate entropy.
- ▶ Pick word with highest entropy.

This approach ensures the most informative guess.

Code Implementation

```
1 static float calculate_entropy(int* pattern_counts) {
2     float entropy = 0.0f;
3     for (int i = 0; i < PATTERN_COUNT; i++) {
4         if (pattern_counts[i] > 0) {
5             float p = (float)pattern_counts[i] / solution_count;
6             entropy -= p * log2f(p);
7         }
8     }
9     return entropy;
10 }
11
```

Code for Entropy Calculation

Strategy Essentials

Optimal guess \rightarrow Entropy & Probability.

Weighted Entropy for Global Optimality

Local entropy maximization may not guarantee global optimality.
Thus, we weight future entropy for a more informed guess.

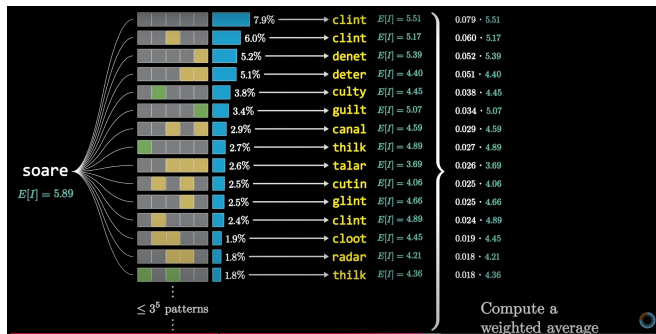


Figure: Image source: 3blue1brown

Decision-Tree-Based Player



```
1 salet BBBBB1 courd BBBBB2 nymph BBBBY3 whiff GGGGG4
2 salet BBBBB1 courd BBBBB2 nymph BGYYB3 pygmy GGGGG4
3 salet BBBBB1 courd BBBBB2 nymph BYBBB3 fizzy GGGGG4
4 salet BBBBB1 courd BBBBB2 nymph BYBBB3 fizzy YGBBG4 jiffy GGGGG5
5 salet BBBBB1 courd BBBBB2 nymph BYBGY3 hippy GGGGG4
6 salet BBBBB1 courd BBBBB2 nymph BYBYB3 piggy GGGGG4
7 salet BBBBB1 courd BBBBB2 nymph BYGGB3 wimpy GGGGG4
8 salet BBBBB1 courd BBBBB2 nymph GGGGG3
9 salet BBBBB1 courd BBBBB2 nymph GYBBB3 ninny GGGGG4
```

Figure: Decision Tree

1. Initial Move:

- ▶ Always starts with "SALET" as the first guess.

2. Pattern Tracking:

- ▶ Maintains a cumulative pattern of guesses and feedback.
- ▶ Format: <WORD> <PATTERN><LEVEL> ...
- ▶ Example: SALET GYBBG1 CRANE GBBBY2

3. Decision Making:

- ▶ Consults `tree.txt` for the next optimal move.
- ▶ Chooses the statistically best next word.

Core Function

```
1 static const char* find_next_move(const char* feedback, int level) {
2     // Append new pattern to cumulative pattern
3     char new_pattern[32];
4     sprintf(new_pattern, "%s %s%d ", current_word, feedback, level);
5     strcat(cumulative_pattern, new_pattern);
6
7     printf("Looking for pattern: '%s'\n", cumulative_pattern);
8
9     // Search for matching line
10    for (int i = 0; i < line_count; i++) {
11        if (strstr(decision_lines[i], cumulative_pattern) == decision_lines[i]) {
12            // Extract next word
13            const char* line = decision_lines[i] + strlen(cumulative_pattern);
14            char next_word[WORD_LENGTH + 1];
15            if (sscanf(line, "%5s", next_word) == 1) {
16                printf("Found next word: %s in line: %s\n", next_word, decision_lines[i]);
17                return strdup(next_word);
18            }
19        }
20    }
21
22    printf("No matching move found\n");
23    return NULL;
24 }
```

Figure: function:find_next_move

Compare Guess Distribution

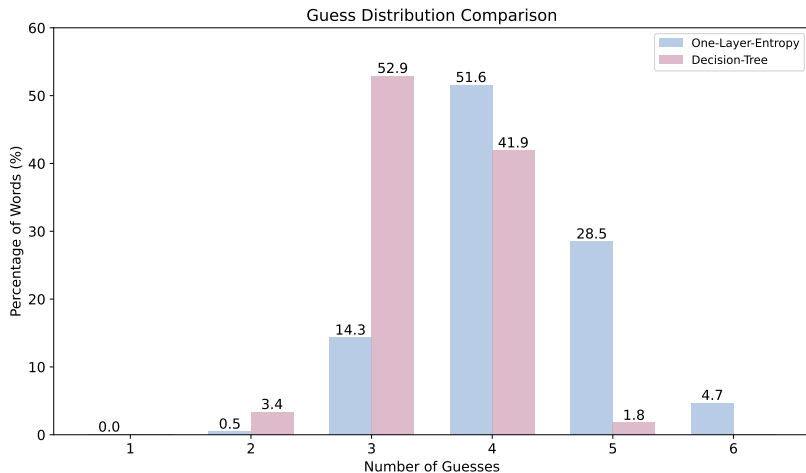
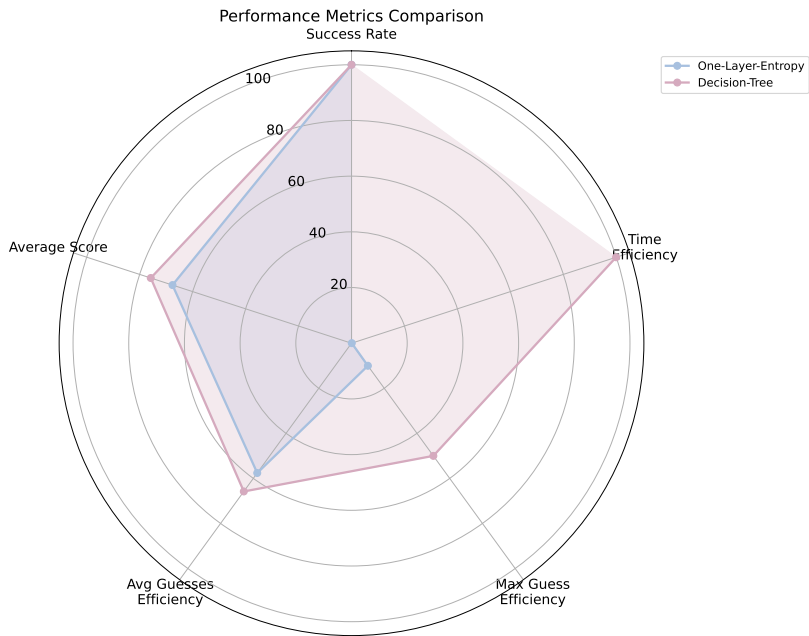


Figure: Compare Guess Distribution

Performance Metrics




References

3Blue1Brown n.d.(b). 3Blue1Brown n.d.(a). Selby n.d.

 3Blue1Brown (n.d.[a]). *Mistake Clarification*. URL:
<https://www.youtube.com/watch?v=fRed0Xmc2Wg>.

 — (n.d.[b]). *Solution to a Math Problem*. URL:
<https://www.youtube.com/watch?v=v68zYyaEmEA>.

 Selby, Alex (n.d.). *The Best Strategies for Wordle*.
https://sonorouschocolate.com/notes/index.php?title=The_best_strategies_for_Wordle. Accessed:
2024-11-26.

Use of AI

Kimi:

- ▶ **Prompt1:** How to solve Wordle and give me some ideas.
- ▶ **Prompt2:** Please refactor my code and add some annotation.