Wordle with Solver

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

Group 1

November 27, 2024

Contents

1

Code Accessibility

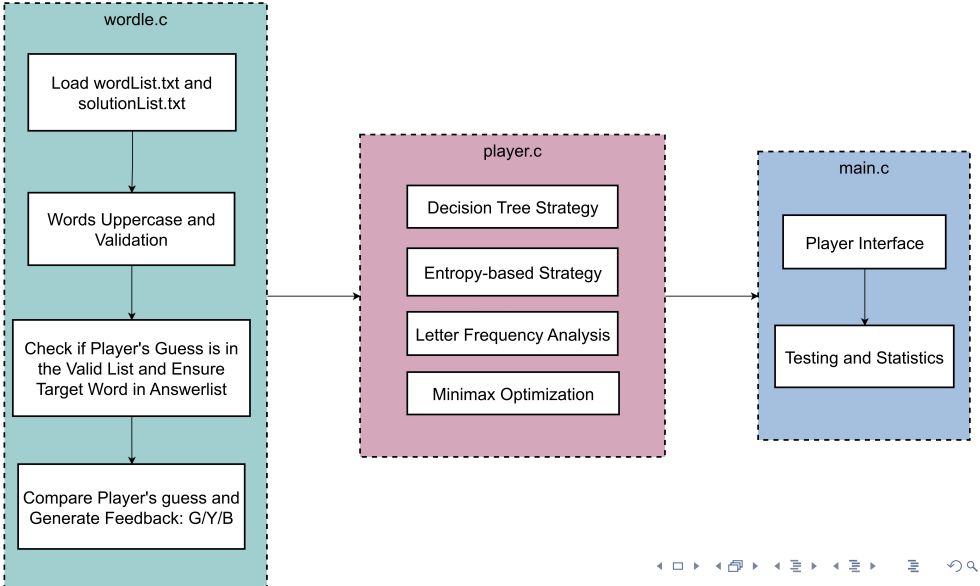
Code Accessibility

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🔗.
DO NOT COPY!
We will change the visibility after presentation.

2

Overview

Overview


wordle.c
Load wordList.txt and solutionList.txt
Words Uppercase and Validation
Check if Player's Guess is in the Valid List and Ensure Target Word in Answerlist
Compare Player's guess and Generate Feedback: G/Y/B
player.c
Decision Tree Strategy
Entropy-based Strategy
Letter Frequency Analysis
Minimax Optimization
main.c
Player Interface
Testing and Statistics

3

Background

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.

4

Minimax-Based Player

Minimax-Based Player

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

$$M(w) = \max_i N_i(w) \tag{2}$$

PLayer_minimax() in Player.c

```
for (int i = 0; i < wordCount; i++) {
    generate_pattern_counts(wordList[i], pattern_counts);

    int max_remaining = 0;
    for (int j = 0; j < PATTERN_COUNT; j++) {
        if (pattern_counts[j] > max_remaining) {
            max_remaining = pattern_counts[j];
        }
    }

    if (max_remaining < min_worst_case) {
        min_worst_case = max_remaining;
        strcpy(best_guess, wordList[i]);
    }
}
```

5

Frequency-Based Player

Frequency-Based Player

$$\text{score}(w) = \sum_{j=1}^{k} (2 \cdot \text{letter\_freq}[L_j][j]) + \sum_{L \in w \text{ (unique)}} \text{total\_freq}[L] \tag{3}$$

$$w_{\text{best}} = \arg\min_{w} M(w) \tag{4}$$

Player_frequency() in Player.c

```
for (int i = 0; i < wordCount; i++) {
    float score = 0.0f;
    int used[26] = {0};
    for (int j = 0; j < WORD_LENGTH; j++) {
        int letter = wordList[i][j] - 'A';
        score += letter_freq[letter][j] * 2.0f; // Position-specific
            score

        if (!used[letter]) {
            score += total_freq[letter]; // Overall letter frequency score
            used[letter] = 1;
        }
    }
    if (score > best_score) {
        best_score = score;
        strcpy(best_guess, wordList[i]);
    }
}
```

6

Entropy-Based Player

Understanding Information Entropy

1. Information Entropy Calculation Formula

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

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.

calculate_entropy() in Player.c

```
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 }
```

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.

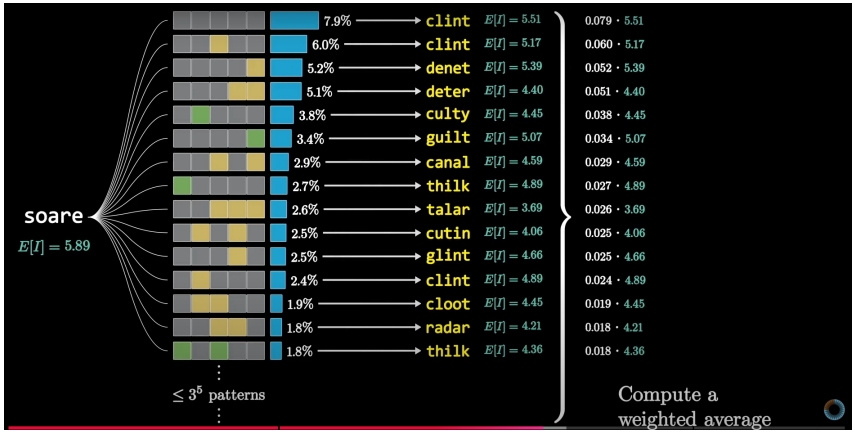


Figure: Image source: 3blue1brown

7

Decision-Tree-Based Player

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

Code Implement

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.

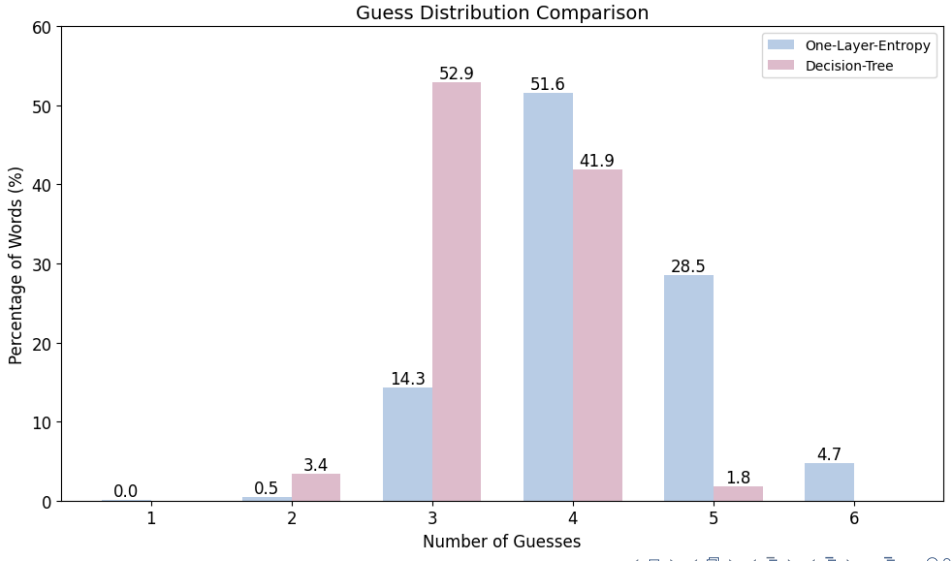
find_next_move() in Player.c

```c
static const char* find_next_move(const char* feedback, int level) {
    char new_pattern[32];
    sprintf(new_pattern, "%s %s%d ", current_word, feedback, level);
    strcat(cumulative_pattern, new_pattern);
    for (int i = 0; i < line_count; i++) { // Search for matching line
        if (strstr(decision_lines[i], cumulative_pattern) ==
            decision_lines[i]) { // Extract next word
            const char* line = decision_lines[i] + strlen(
                cumulative_pattern);
            char next_word[WORD_LENGTH + 1];
            if (sscanf(line, "%5s", next_word) == 1) {
                printf("Found next word: %s in line: %s\n", next_word,
                    decision_lines[i]);
                return strdup(next_word);
            }
        }
    }
    return NULL;
}
```

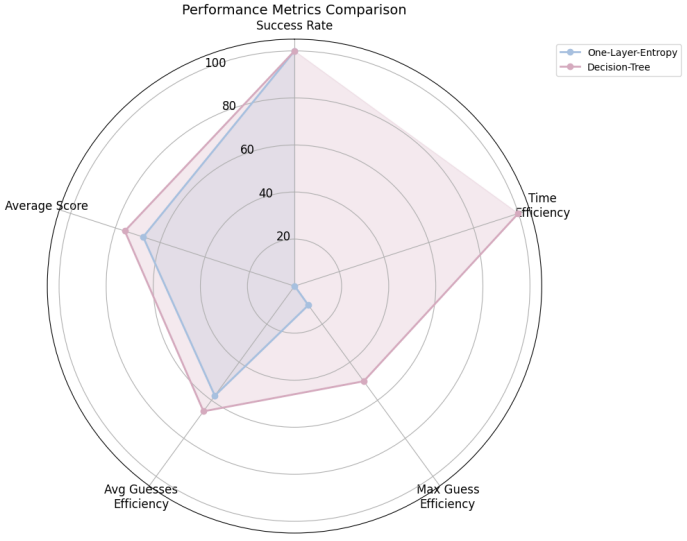
8

Result

Compare Guess Distribution



Performance Metrics



9

References

References I

[1] 3Blue1Brown. Solution to a Math Problem[EB/OL]. https://www.youtube.com/watch?v=v68zYyaEmEA.

[2] 3Blue1Brown. Mistake Clarification[EB/OL]. https://www.youtube.com/watch?v=fRed0Xmc2Wg.

[3] SELBY A. The Best Strategies for Wordle[Z]. https://sonorouschocolate.com/notes/index.php?title=The_best_strategies_for_Wordle. Accessed: 2024-11-26.

10

Use of AI

Use of AI

Kimi

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

Thank You