Wordle

Introduction to AI Final Project

School of Computer Science and Engineering
Hebrew University of Jerusalem

Adam Shtrasner Ofri Shtaif Yuval Hefets Shelly Madar

Abstract*—In this project our aim was to solve the game of Wordle using the following AI methods: Adversarial Search approach: min-max, alpha-beta and expecti-max search; Decision Tree approach: decision tree. All of those while applying different heuristics, and analyze their performance.

Min-Max, Alpha-Beta Pruning, Expectimax, Decision Tree, Wordle

I. Introduction

Since its release in October 2021, the game of Wordle has gained a large amount of popularity. The idea behind the game is quite simple: each day, a 5-letter word is chosen, and the player needs to guess the word within 6 tries. If the player fails to do so – he loses. For each try, the game gives feedback, which aims to help the player on their next try. The feedback is given by marking each guessed letter with one of 3 colors:

- Green, indicating the letter is correct and in the correct position.
- Yellow, indicating the letter is correct but is not in the correct position.
- Grey, indicating the letter is not in the answer at all.

For example, if the answer is "arose", and our guess is "aback", the letter 'a' will be marked as green because the first letter of both of the words is 'a', while the rest of the letters do not match the answer, and so they will be marked as grey. Notice how the 'a' in the 3rd position of our guess is not marked yellow, since there is only one 'a' in the answer word, meaning repeating letters will be marked as green or yellow only if they appear in the guessing word the same amount as in the target word.

The main goal of our project is to build an AI agent which would solve the game as fast as possible in terms of the amount of guesses. Also, we found out that while using the Entropy heuristic (specified in the Heuristics section), there is no agent that can solve the game in less than 3 tries on average [1], so we know that we cannot do better than that.

II. Approach and Methods

Our approach was to look at the game as if there are 2 players, and so we established a state space, and modeled our problem as specified below.

A. State Representation

We first defined the problem as a search problem, where each node consists of: the guessed word, and the list of all possible words to guess.

B. Adversarial Game Point of View

Notice that this is a one player game. Then how come we use adversary game methods to solve it you might ask? Well, as we have stated earlier, for each guess there is feedback from the game. Therefore, we can look at the game as if two players are playing: one is trying to guess the right word – the Guesser, and the other is giving us feedback according to the target word – the Indicator.

The Guesser's action is a word to guess out of the current possible words list (at each guess, the list of possible words is narrowed down according to the indication).

The Indicator's action is an indication out of the list of possible indications a word can have.

C. Algorithms

We decided to solve the game using four search algorithms:

- **Mini-Max:** The guesser is trying to maximize the evaluation function, while the indicator is trying to minimize it.
- **Alpha-Beta Pruning**: Same as mini-max, but this algorithm will decrease the number of nodes that are evaluated by the minimax algorithm in its search tree.
- **Expecti-Max:** The guesser is still trying to maximize the evaluation function, while the indicator takes the average of all the evaluations of the nodes on a specific level.
- **Decision Trees:** The guesser is trying to maximize the evaluation function, but without going "deeper" in the search tree, only wisely choosing a guess out of the currently possible words.

III. Heuristics

A. Entropy

Entropy is a concept from Information Theory, which basically means the average amount of information conveyed by an event [2]. In our case, we would like to evaluate how much information we get from a word based on all possible indications. For each word w, we calculate the entropy as follows:

$$H(w) = \sum_{x} p(x|w) \cdot \log_2\left(\frac{1}{p(x|w)}\right)$$

Where x is an indication out of all the possible indications, and $p(x|w)$ is the probability of x being the indication on w. In particular, $p(x|w)$ is calculated as follows:

let n be the number of words that satisfy the indication x according to a given word w, and let N be the total number of possible words. Then:

$$p(x|w) = \frac{n}{N}$$

In our code, when a state has more than THRESHOLD_EVAL_FUNC possible indications, we chose to use the constant evaluation function (explanation on the Evaluation Functions section), because when the number of possible indications is too high, the computation of the entropy is very expensive in terms of runtime. Our choice for THRESHOLD_EVAL_FUNC was 100 indications.

B. Average Number of Green/Yellow/Grey letters in a Word

For each word w, we counted the number of green, yellow and grey letters it would get on each of the other words if it were the target word. This way, we took the average amount of green, yellow or grey letters each word would get.

IV. Evaluation functions

We chose 3 evaluation functions, the first 2 are based on the same calculations, combining the heuristics stated above with weights. Denote:

Entropy(w) – The entropy of a word w.
avg(green_letters|w) – Average of green letters in a word.
avg(yellow_letters|w) – Average of yellow letters in a word.
avg(grey_letters|w) – Average of grey letters in a word.

Then the calculation is as follows:

$$0.6 \cdot Entropy + 0.2 \cdot avg(green\_letters) + 0.15 \cdot avg(yellow\_letters) + 0.05 \cdot avg(grey\_letters)$$

A. Real-Time Evaluation (Local)

All of the parameters are calculated based on the state's current possible words and possible indications.

B. Constant Evaluation (Const)

All of the parameters are calculated in advance based on all of the words and all of the indications.

C. Zero-Evaluation

This evaluation functions returns a score of zero to each state, which basically means choosing a random word out of all possible words.

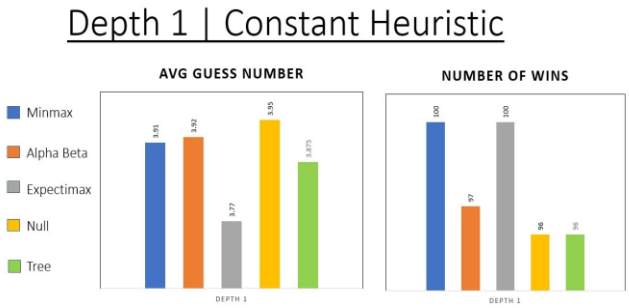
V. Implementation

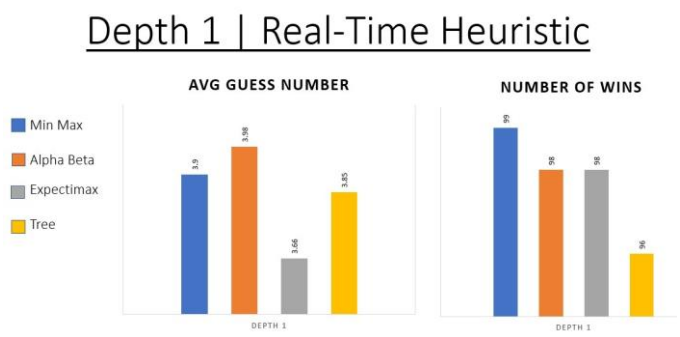
Originally, the game is implemented in JavaScript (by Josh Wardle), and there are 12,972 possible words that the player can guess, while only 2309 of them can be the answer. But, we have decided to implement our own version of the game in Python, where the list of possible words to guess is narrowed down to 3000 most frequent words out of the 12972 possible words. The reason for choosing 3000 words is because we assumed that a proper game would choose only frequent words in English. Moreover, going over all of the 12972 words is too expensive in terms of run time.

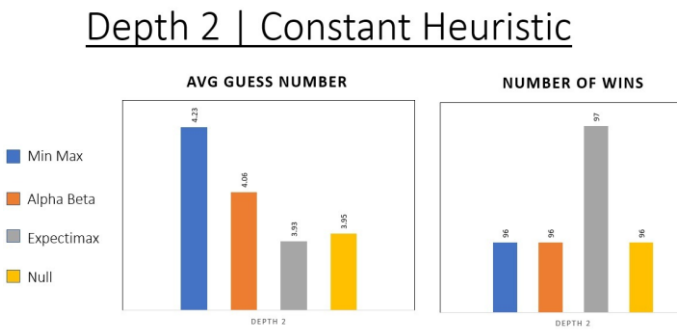
Also, when choosing to play with the agents, our first guess will always be the word "CARES", since it has the maximum value of the calculation specified in the section above, meaning this word gives us in average the most information, and so it would be intuitive to use it as a first guess.

VI. Results

We tested the performance of each agent on 100 runs on average, and got the following results:

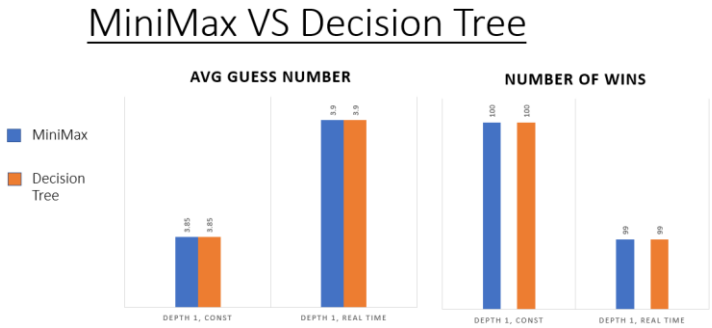






It is clear that the Expectimax Agent was able to guess the right word with the minimum amount of guesses, and we actually anticipated that, since Wordle is based on random moves. Also, with depth > 1 there was no point in testing MinMax and AlphaBeta agents with real-time evaluations, since their decisions are then made according to specific indications, other than Expectimax. On depth 2 and real-time evaluation, Expectimax made 3.71 guesses in average, and successfully guessed 97 words out of 100.

Also, when comparing the performance of the decision tree agent with the MinMax agent with depth 1, we get the same results overall, meaning that applying the decision tree agent is equivalent to applying the MinMax agent with depth 1:



We therefore conclude that the best agent to solve the game in terms of the average number of guesses and the number of wins is the ExpectiMax agent.

REFERENCES

[1] 3Blue1Brown, Solving Wordle Using Information Theory: https://www.youtube.com/watch?v=v68zYyaEmEA, 29:10 – 29:55

[2] Entropy Wikipedia Page: https://en.wikipedia.org/wiki/Entropy_(information_theory)