Wordle AI

Introduction

Wordle is a web-based word game that went viral in 2021. Players have six guesses to try to guess the five-letter word of the day. After each subsequent guess, the player is given feedback on which letters in the guessed word are in the correct position (shown as green), are in the word but incorrect position (shown as yellow), or not in the word at all (shown as gray). Failure in guessing the word of the day within six guesses results in a loss for the day. There have been many similar game applications that arose after the release of Wordle.

Additionally, there are two modes in which Wordle can be played - easy mode and hard mode. In easy mode, players can guess whatever words they would like to. In hard mode, if players have guessed letters correctly at all (regardless of whether or not the letter is in the correct position), their subsequent guesses must be words that also contain those letters. Our solutions will solve the game in easy mode and adopt the list of all five-letter words and all possible five-letter answers sourced from Wordle.

Our goal is to develop an AI to solve Wordles and minimize the number of guesses required to solve the challenge. We wanted to make the game as user front-facing and as interactive as possible. This meant that the user would be able to interact with our Wordle AI and play on the platform. Hence, we decided to develop a website application to host the Wordle game which is integrated with the algorithms. The AI that we developed attempts to suggest the best twelve words to use on each guess stage, six for each of the two algorithms that we implemented, learning from previous guesses and minimizing the number of turns it takes to select the correct word.

We developed two different algorithms to minimize the number of guesses required to win and evaluated their average performance to decide the optimal strategy to adopt. We used metrics including the average number of guesses to solve the challenge and the average probability of winning the game in six guesses to evaluate our algorithm

Prior Work

Because of the viral nature of Wordle in early 2022, there have been extensive efforts to find an optimal algorithm to guarantee a win or to find the answer with minimal guesses. Below are just a few popular strategies that people have tried and tested.

Tyler Glaiel Algorithm

Glaiel's algorithm gives a hypothetical score to each word in the list of possible guesses (Glaieil). The algorithm behind giving that hypothetical score is found by checking every possible guess against every possible solution. Desirable green letters are worth two points, less certain yellow ones are worth one point, and the useless gray ones are worth zero points. At each step, the guess with the highest average score is returned as the guess to be made. In his study, he found that:

1. SOARE: This word had the highest score among all the potential guesses in Wordle. On average, it finds the solution with 3.69017 guesses. However, this was often not fast enough to help players get to the solution; it takes 8 guesses to find the solution in the worst-case scenario.
2. ROATE: This is the word that was able to get to solutions the fastest. On average, it takes 3.49417 guesses to find the solution. Also, it has a worst-case of 5 guesses.

Mahmood Hikmet Algorithm

Hikmet designed a Wordle solver tool called Unwordle to help solve the puzzle with a 99.3% success rate (Hikmet). He came up with an algorithm that works by deducing the frequency at

which certain letters appear in the list of possible solutions and ignoring other guesses. The words that use the most popular letters in the list of solutions are considered the best starting words to guess. After each guess, certain words are eliminated from the list of possible solutions; therefore, the list keeps converging with each guess. It basically examines the list of possible answers in Wordle to figure out the best possible starting guesses.

Andrew Taylor Algorithm

Taylor's algorithm essentially combines the above two algorithms. It assigns a numerical score to each potential Wordle solution, with the lowest score signifying the best solutions worth guessing (Taylor). With this algorithm, he found that:

1. REAIS: Taylor named this as the best starting word, with only 168 potential solutions having none of those letters.
2. BLAHS: Taylor named this the best word for eliminating most of the answers.
3. CENTU: This was the second-best word for eliminating most of the answers.
4. DOGGO: This was the third-best word for eliminating most of the answers.

Greedy Min-Max and Average Minimize

1. Greedy Min-Max: Choose the word which minimizes the worst-case length of the narrowed-down answer list after making a guess.
2. Average Minimize: Choose the word that minimizes the average-case length of the narrowed-down list (Interview).

Exact Dynamic Programming

Developed by two Operations Research professors from MIT, this algorithm has one of the best average win rates amongst all existing algorithms (Bertsimas). The crux of the algorithm comes from a series of theoretical and computational observations from the Bellman Equation solving the game. It creates a formulation of the game into a tractable problem with a finite

state-space, by outlining the definition of a state s, control a, cost function c, and state transitions of the underlying Markov Decision Process (MDP). Given this formulation of Wordle as a Markov Decision Process, the algorithm applies the Bellman equation to solve for the optimal value of a state shown below:

$$V_t^*(s_t) = \min_{a \in A} \left\{ c_t(s_t, a) + \sum_{s_{t+1} \in P(s_t, a)} p_t(s_{t+1}; s_t, a) V_{t+1}^*(s_{t+1}) \right\},$$

or more explicitly:

$$V_t^*(s_t) = \begin{cases} 1, & \text{if } t < 6, |s_t| = 1, \\ \min_{a \in A} \left\{ 1 + \sum_{s_{t+1} \in P(s_t, a)} p_t(s_{t+1}; s_t, a) V_{t+1}^*(s_{t+1}) \right\}, & \text{if } t < 6, |s_t| > 1, \\ \infty, & \text{if } t = 6. \end{cases}$$

This model solves the game in just two guesses 4% of the time and in three guesses 57% of the time. It only needs a fifth guess 3% of the time, and it never relies on that sixth and final round. Overall, it takes an average of 3.421 guesses to reach the answer.

Methods

Website Application

In order to create a website application, we use React.js as our client-side JavaScript frontend framework, Express.js as our backend server-side website framework and Node.js as our JavaScript server platform. The algorithms are written in Python and are run as a script on the backend when prompted.

On the frontend, we implemented all functionalities of the Wordle game which track the state of the game and reflect the state on the UI accordingly. The styling of the normal Wordle board is taken from open source code (Kubów) with a couple of changes to suit our purposes. Upon restarting a new game, the frontend sends a HTTP request to the server and Express.js handles the request. The server receives the HTTP requests and spawns a child process to run the python script where the algorithm is written. The python script receives the game state such as all previous guesses and the correct Wordle answer as parameters from the server. The algorithms then generate word suggestions based on the game state which get parsed back to the server and server sends the word suggestion data including word suggestions and metrics data to frontend as a HTTP response.

We included a guarantee win feature (algorithm described under the “Algorithms: Guarantee Win” section) which can be used to check if the selected word that the player is considering to guess has a path to win the game in later rounds. The feature is only enabled on the 4th to 6th stage. Once the player hits the fourth guess, they can input a word without confirming it as their fourth guess and click on the “check” button to check if there are guaranteed paths to win by using the word at that round. A message would show on top of the Wordle board: either “Guaranteed will complete in 6 guesses” or “Not guaranteed will complete in 6 guesses” to indicate the output of the guarantee win algorithm.

Additionally, we indicated whether a word suggestion is a possible answer on the UI to provide the player with more information when they are choosing a word suggestion as a guess to make. This is done by checking all word suggestions if they belong to the remaining sample space set which contains all words that are possible to be the correct answer. The information is then sent from the Python script through our backend server to frontend client-side as binary indicators. The information is then displayed as a border around the word suggestion. We added a legend “Green bordered suggestions are possible answers.” to ensure that the player understands the meaning of the green borders.

Graphic Interface

The graphical interface is kept clean and simple. The left side contains the normal Wordle game while the right side contains the suggested words from the two algorithms described below. The suggested words can be clicked to autofill the Wordle for convenience, and there is a loading spinner that appears when the words are being fetched after each guess as a visual indicator because it takes a few seconds for the algorithms to run. Additionally, suggested words that are surrounded by a green box are words that are possible Wordle answers. Finally, a guess button to check a guaranteed win appears next to each row in the Wordle board after three guesses have been made and the row is filled out. We made this decision because the guarantee win algorithm takes a long time when the search space is too large.

Word Lists

In order to make sure that all accepted guesses are valid 5-letter words, we will need to compile a list of valid 5-letter words. There are around 13,000 of such words, and there are two possible ways that we came up with to retrieve the list: (1) Use the official Wordle library, and (2) Use the nltk corpus and filter through all words to get words that have 5 letters only. We ended up using (1) to retrieve all allowed 5-letter words in the dictionary to better reflect on the Wordle game settings. The allowed 5-letter words are valid 5-letter words that are acceptable as a guess in Wordle. We further obtained all possible 5-letter words from the official Wordle library which contains all words that are acceptable as answers. The word list with possible words (2309 words) in *wordle-dictionary.js* is a subset of the word list with all allowed words in *dictionary.js*.

Pattern

In order to speed up the process of obtaining possible words given the information at each stage, we precomputed and cached a giant look up table containing a pattern between two words for all possible pairs of words. Formally, we produced a 2D table D such that

D[word1][word2] = pattern that would be obtained if we guessed word1 but the answer was word2.

To encode the pattern, the colors are represented as numbers: 0 for gray, 1 for yellow, and 2 for green. After obtaining the 5 digit sequence, we consider it as a base-3 number and convert it to ternary. As a result, patterns are ternary numbers in the range [0, 242].
The implementation is a modification of 3b1b's implementation (3b1b).

Algorithms: Maximizing Entropy

Our main approach is building an AI solver based on information theory. At each word guess, we calculate entropy, or information gain, for all allowed 5-letter words by inferring from the blocks of green, yellow, and gray squares on previous guesses that reveal the letters the player got right or almost right. Then the words with the highest expected information will be selected as the next suggested guesses. The process is done on all allowed 5-letter words that are accepted by Wordle instead of only the possible 5-letter words that are not yet eliminated to maximize information scores. Since this approach maximizes entropy, we understand that the provided options at each stage do not always need to include the correct answer - sometimes it is a better strategy to sacrifice using a guess by guessing an impossible word (word that is not possible to win) to gain more information about the correct answer. For example, the algorithm can try to maximize information by suggesting words that do not have repeating letters as the first guess for the second guess. So instead of using the inference from the blocks of green, yellow, and gray squares from the first guess to make a second guess, the algorithm could produce a second guess that has no overlapping letters as the first guess to maximize information about the answer.

Algorithms: Maximize Green Letters

Our second approach chooses words that maximize the probability of getting the most green letters. For the set of all possible Wordle answers left, look at each letter position of the word

and check the number of possible Wordle answers with the same letter at the exact position. The algorithm sums up the number of letter matches to score each word. The UI will then display six words with the highest scores.

For example, if the algorithm is looking at the word “where”, then the letter mapping could look something like this:

1. “w”: {“which”,“whale”, “weary”},
2. “h”: {“which”,“whale”, “chalk”},
3. “e”: {“break”},
4. “r”: {“weary”, “carry”, “curry”},
5. “e”: {“whale”}

for a total score of eleven. This algorithm chooses the optimal words that maximizes the probability of having correct letters in all letter positions.

Algorithms: Guarantee Win

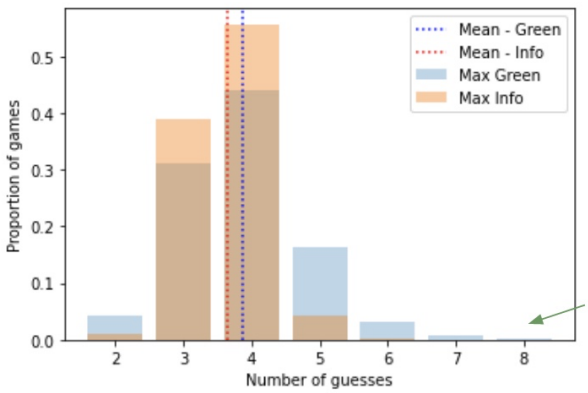
We provided an additional tool for users after round 3 to check if there is a possibility of not being able to finish the game with the remaining number of guesses given a particular word choice. This is a recursive algorithm with the base case: if there is only one remaining guess, then winning (i.e. finishing the game in this guess) is guaranteed if and only if there is one possibility left. For more than one round left, the algorithm searches recursively, for each pattern that can be produced by a particular guess, if there is at least one choice in the next round that can guarantee win. Winning is only guaranteed if this condition is met for every single pattern that can possibly be generated between the selected word and all remaining possible words. To use this feature, the user would have to check a word choice at every round to make sure they are following a “safe” path, until they ultimately obtain the answer.

Results

Test results

Statistics – simulation for all 2309 possible words

	Max Info	Max Green
Mean	3.636	3.866
Probability of losing	0	0.996%
Probability of <=2	0.953%	4.2%
Probability of <=3	40%	35.3%



Extreme cases, which can be attributed to the case where there are substantial green letters but remaining sample space is still huge

We ran the two algorithms on all possible 2309 words Wordle would pick. The overall performance for max info is better, which is expected. It has a mean of 3.636 guesses, and even more encouraging is the fact that it never uses more than 6 guesses, meaning it is always able to complete the game. What's worth noticing for max green is that it performs better in terms of being able to get the answer in two trials. This might be beneficial if one's objective is

to get the answer using as few guesses, even though it does not happen extremely frequently.

A Closer Look at the Worst Cases

soare	soare
talar	talar
pigmy	pigmy
wrick	wrick
zurfs	zurfs

Didn't get down to rarer letters like "w" and "z" until later rounds

	WAFER	WAVER	GONER	MAMMY	VAUNT	WATCH	WIGHT
Max Info	6	6	5	5	4	4	4
Max Green	3	5	8	8	8	8	8

Once identified several green letters, just try the more common ones containing them, ended up disastrous for the least common word for the given combo

saree	saree	saree	saree	saree	saree
water	rider	canny	canny	canny	bonny
wafer	cower	badly	taunt	match	chich
wager	hover	tatty	daunt	batch	fight
	joker	gawky	gaunt	hatch	light
	boxer	happy	haunt	latch	might
	foyer	jazzy	jaunt	patch	tight

Looking at the worst case performance provides us with more intuition into what the algorithms tend to do. Here is a list of words the two algorithms performed most poorly on respectively. For max info, "WAVER" and "WAFER" contain letters like "W", "V", "F" which are less common especially in the given position. Therefore, the algorithm didn't start guessing words containing them until much later. On the other hand, since the green algorithm knows the word is "wa_er" since the 2nd round, it takes no more than 5 rounds to just try all the remaining possibilities from most (green algorithm's definition of) "probable" to least. This is a demonstration of why we might want to consider using this algorithm if the remaining number of possibilities is small.

As for max green, it is obvious that once it identified several green letters, it just kept tying words containing such patterns, from the more common letters to the less common letters. In the guessing sequence for "GONER", for example, the algorithm knows after the third round that the word looks something like "_o_er", so it guessed words containing it, starting from

“hover”, possibly because “h” is a more common start for such a pattern, then it goes down to a less common “j” start, then “x” and “y” in the middle which are far less common.

Discussions

Our main contribution comes from the semi-systematic approach we have taken – instead of writing a fully automated bot, we decided to instead give more intuition behind each algorithm as well as the game state. That way, not only does the user get a better understanding of the mechanics behind the algorithms, but they also have the freedom to adapt the algorithms in a way that fits their objectives.

Considering the possibility of different objectives a user might want to optimize, we also evaluated our results holistically – apart from the average number of guesses, which is what most prior work reported, we also considered other metrics such as the distribution of the guesses, as well as best case and worst case performance. By analyzing special cases in detail, we again gained a deeper understanding of the strengths and weaknesses of each, and this information would hopefully provide intuition into which algorithm is desirable under specific situations.

Conclusion

In this project, we not only designed and implemented two algorithms for playing Wordle, we also developed an interface which provides insight into why each word is good and lets users freely choose what they want to choose. We also included additional tools such as displaying the number of possibilities left, and letting users check if a word choice is “safe” in the sense of whether there is a possibility that they might not be able to finish the game.
One interesting future direction would be to look at how a combination of these two strategies will perform. If we were to make it a fully automated bot, we could either decide on some rules as to when to use which algorithm (potentially depending on the remaining number of possibilities), or, even more interestingly, use a weighted approach to get the final scoring

metric. The weights can be fixed or as a function of the number of rounds, remaining number of possibilities, and number of green letters guessed, or determined through optimizing some custom objectives. The idea is that we would consider a higher weighting on choosing the maximizing entropy strategy on the first one or two rounds, and increase the weighting on the green letter strategy on the latter rounds, also depending on the remaining sample space size and the number of green letters guessed right.

References

Bertsimas, D. and Paskov, A., "An Exact and Interpretable Solution to Wordle," 2022. https://auction-upload-files.s3.amazonaws.com/Wordle_Paper_Final.pdf

Glaiel, Tyler, “The mathematically optimal first guess in Wordle,” Dec 29, 2021. Medium. https://medium.com/@tglaiel/the-mathematically-optimal-first-guess-in-wordle-cbcb03c19b0a

Hikmet, Mahmood, “The Best Wordle Strategy + Free Tool,” Jan 4, 2022. Youtube. https://www.youtube.com/watch?v=hJJaYvxQh8w&ab_channel=MahmoodHikmet

Taylor, Andrew, “How to Always Win at Wordle,” Jan 6, 2022. Youtube. https://www.youtube.com/watch?v=Xv7JBbOiBkI&ab_channel=AndrewTaylor

Interview Kickstart Team, “What Is Wordle and How to Create a Wordle Solver,” February 8, 2022. Interview Kickstart. https://www.interviewkickstart.com/blog/what-is-a-wordle-solver

Kubów, Ania, “Wordle-Javascript,” https://github.com/kubowania/wordle-javascript/blob/main/style.css

3b1b, “Wordle,” https://github.com/3b1b/videos/tree/master/_2022/wordle