Wordle Turtle: A Wordle Bot

Ava Schneider and Ellery Bock

Abstract

The following project is an implementation of Wordle and a solver (the Wordle Turtle). The Wordle game is playable and uses the real Wordle words. The approach we have taken is to treat Wordle as a Constraint Satisfaction Problem (CSP) and solve it using CSP techniques. We will create two baselines, a random choice baseline, and a word elimination baseline. We will then create a heuristic-based CSP solver and compare all results.

The purpose of the random choice baseline is to show the unlikelihood of solving Wordle with no processing of the feedback from the game. Our results show that the random choice baseline is unable to guess the word within six guesses the majority of the time. The word elimination baseline resembles how a human would play the game of Wordle. Our results show that most of the time the elimination baseline can solve the Wordle within 6 guesses.

Our heuristic-based CSP solver was also able to solve the wordle within 6 guesses the vast majority of the time. Results show that while the elimination baseline and the CSP solver have a similar win rate, the CSP solver significantly reduces the average guess count.

1 Introduction

Wordle is a daily word game that is owned by the New York Times and is a staple in their daily game suite. It is played by hundreds of thousands of people every day, including ourselves [6]. The game of Wordle plays as follows:

1. A 5-letter word is chosen from a list of words. This word is unknown to the player.
2. The player is allowed 6 chances to guess the word by typing other 5 letter words.
3. After each guess, the player gets feedback on their choice. The player is shown a grey square for the letters they guessed that are not in the answer, a yellow square for the letter that is in the answer but not in the place they put it, or a green square for a correct letter in the correct spot [6].

We decided to create a Wordle bot because we both have had a good amount of algorithms/ ML experience from other classes, but haven't had any practice with CSPs. We also both got the CSP question wrong on the exam so we wanted to explore the topic further in this project. We chose Wordle because we both play it every day and have a good understanding of the mechanics of the game.

This paper will explore solving the game of Wordle using a heuristic-based CSP. We will define the domains, the variables, the constraints, and a heuristic. We will then compare the results from the CSP to the results of two baselines, a random choice baseline and a word elimination baseline. By doing this our project aims to answer the following questions:

1. Can CSP techniques effectively solve Wordle?
2. How does the CSP solver compare to baseline models?

2 Related Works

Because Wordle has become so popular some researchers have worked on creating solutions to solve Wordle [5]. Solvers have ranged from entropy to reinforcement learning [3][5]. All of these solvers have the same goal, solve Wordle, but there are a plethora of ways to reach this goal.

Our work differs from other works in two ways. First, we incorporate two different lists, one that contains all possible answers to the Wordle and one that contains all possible guesses you can make. Other research has been done using only the answer word list [4][5]. Using both lists allows us to use words that don't appear in the answer list to gain valuable information about the answer. Without the additional list of possible guesses, we limit the amount of knowledge that can be gained.

The second difference is how we modeled the Wordle solver. We used a heuristic-based CSP that prioritizes words that have higher letter frequencies (scores). Other solvers used entropy [3], Monte-Carlo tree search [4], and reinforcement learning [5].

The Monte-Carlo tree search solver won 84% of games and averaged 4.88 guesses [4]. The reinforcement learning solver had a 98% win rate and averaged 4.13 guesses [5]. These past Wordle solvers give good information into what results we should expect from our solver and give us an additional baseline to meet or surpass.

3 Methods

3.1 Dataset

We used two datasets to create this Wordle solver. One is the list of past Wordle answers (2309 words) [11]. Two, the list of valid guesses (12546 words)[10]. These are different, as the answers are a specific subset of the valid guesses. The answers are never plurals that end in 's' or 'es', never past tense that end in 'ed' [2]. For our purposes the two lists are disjoint. The answers only contain the Wordle answers, and the valid guesses contain all non-answers [10][11].

To preprocess the data we create three data frames, one for the list of answer words, one for the list of guess words, and one of the two combined. Each data frame has the whole word in the first column and each letter alone in the subsequent five columns. We use these data frames to build our heuristic.

3.2 Random Baseline

The random baseline guesses random words from both lists until the final guess where it guesses an answer word.

3.3 CSP: Constraint Propagation

A Constraint Satisfaction Problem (CSP) is a type of solution search where a solution must satisfy a list of constraints [7]. These problems are made up of domains, variables, and constraints [7]. They can be solved in many ways, but we are solving using constraint propagation and a heuristic. Our problem domain is the two lists of words. Our variables are each position in the five-letter word. The variable domains are (initially) the letters A-Z. The variable domains are what will be pruned with constraint propagation. The constraints are represented by the feedback given after each guess. In our implementation the number 2 represents a correct letter in the correct spot, the number 1 represents the letter that is in the answer but not in the correct place, and a 0 represents the letter guessed that is not in the answer.

The Wordle CSP is solved in the eliminate_words() function. This function performs constraint propagation, narrowing down the domains (possible letters) of the variables (each letter position) using the feedback as constraints. The function returns only words that satisfy every constraint. There will always be at least one, the answer. By pruning the search space, the solver incrementally converges toward finding the answer. Our elimination baseline uses only this function.

	1	2	3	4	5
a	140	304	306	162	63
b	173	16	56	24	11
c	198	40	56	150	31
d	111	20	75	69	118
e	72	241	177	318	422
f	135	8	25	35	26
g	115	11	67	76	41
h	69	144	9	28	137
i	34	201	266	158	11
j	20	2	3	2	0
k	20	10	12	55	113
l	87	200	112	162	155
m	107	38	61	68	42
n	37	87	137	182	130
o	41	279	243	132	58
p	141	61	57	50	56
q	23	5	1	0	0
r	105	267	163	150	212
s	365	16	80	171	36
t	149	77	111	139	253
u	33	185	165	82	1
v	43	15	49	45	0
w	82	44	26	25	17
x	0	14	12	3	8
y	6	22	29	3	364
z	3	2	11	20	4

3.4 Heuristic

From our class lectures, a heuristic is a function that estimates how close a state is to a goal and gives problem-specific guidance toward a solution. For our specific problem, the heuristic guides us towards the next best guess. The next best guess will give us the most information to give to our constraint propagation.

We get the next best guess in a couple of steps. First, we calculate the frequency of each letter A-Z for each position of the words in the answer list. For example, “Q” in the 1st position has a frequency of 23, and the 5th position has a frequency of 0. 23 words in the answer list begin with “Q” but 0 words end with “Q”.

Then we give all of the words a score based on the frequency of their letters. For example, the word “SLATE” is the best starting guess for Wordle because its letters are the most frequent in their given positions.
{‘S’ : 365, ‘L’ : 200, ‘A’ : 306, ‘T’ : 139, ‘E’ : 422} These values are summed to give us a score of 1432.

Finally, this is brought together in our function use_heuristic(). From the pruned list of words, instead of making a random choice like in the two baselines, we make a heuristic-informed choice. We choose the word that will give us the maximum amount of information, the one with the most frequent letters at each location, which in turn will prune our problem domain more efficiently.

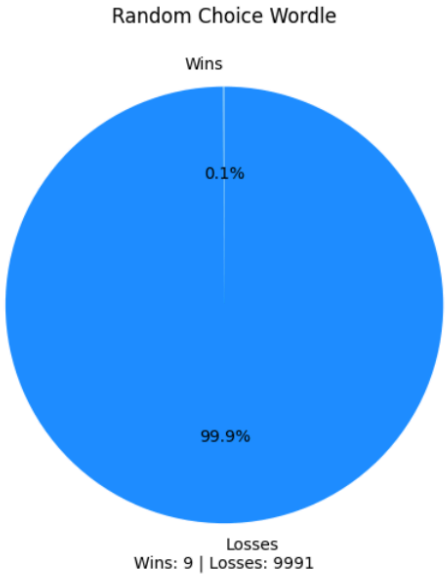
Furthermore, we nudge the solver towards the answer list, as the answer list will contain the goal. We do this in two ways. First, if the number of remaining guesses is more than the number of words in the answer list, we are guaranteed a win by choosing from the answer list only. Second, we use a multiplier to weigh the answer words more heavily. This guides the solver towards guessing answer words at different key parts of the game. We want to guess an answer word to start, to get as much information about the answer list as possible, so a multiplier of 2 will guarantee this. Afterward, we want to use both lists because we can get more general information about the constraints this way. In the 4th and 5th guesses, we nudge a little further toward the answer list, and on the final guess, we make the final hail-mary guess only from the answer list (both baselines use this last guess structure). Our “Wordle Turtle” solver uses this function and the constraint propagation.

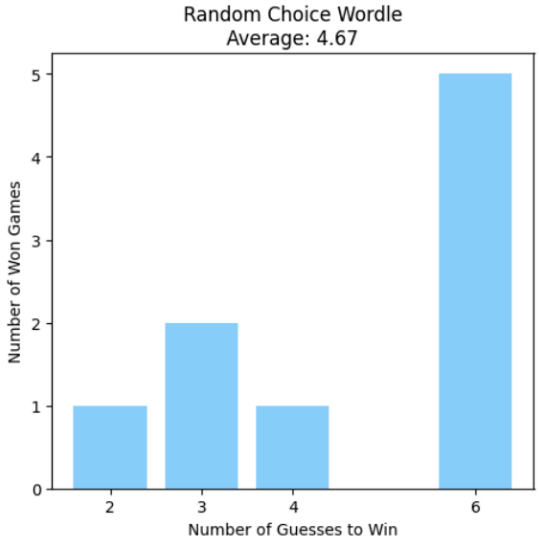
4 Results

We ran each solver 10,000 times per session. The results discussed may vary very slightly due to their random natures, but never enough to change the comparisons between them.

4.1 Random Choice Baseline

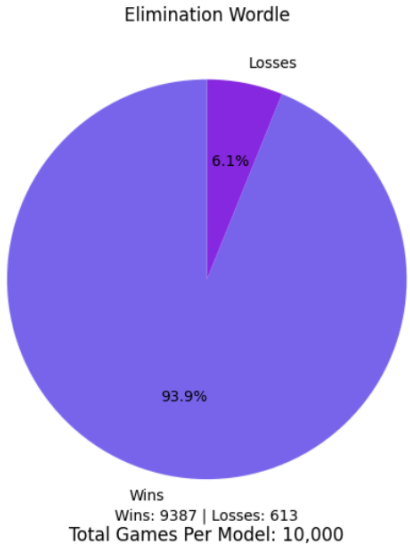
The random baseline performs abysmally as expected. It was able to win Wordle less than .1% of the time.

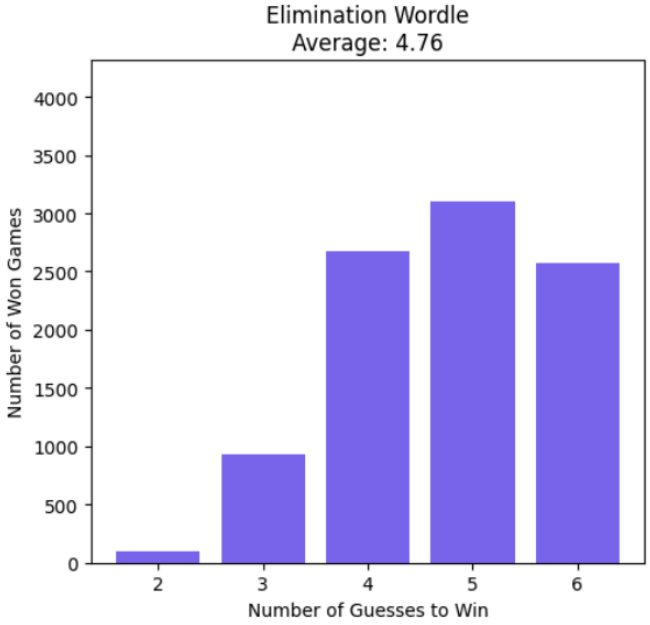




4.2 Word Elimination Baseline

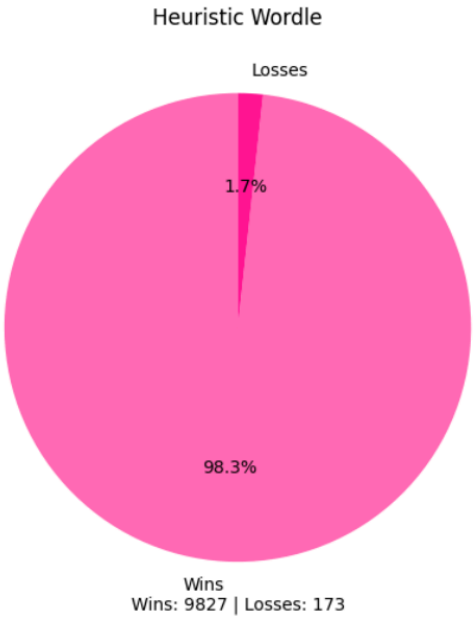
The elimination solver performs very well. It is able to win Wordle 93.9% of the time, with an average of 4.76 guesses. This, of course, beats the random baseline.

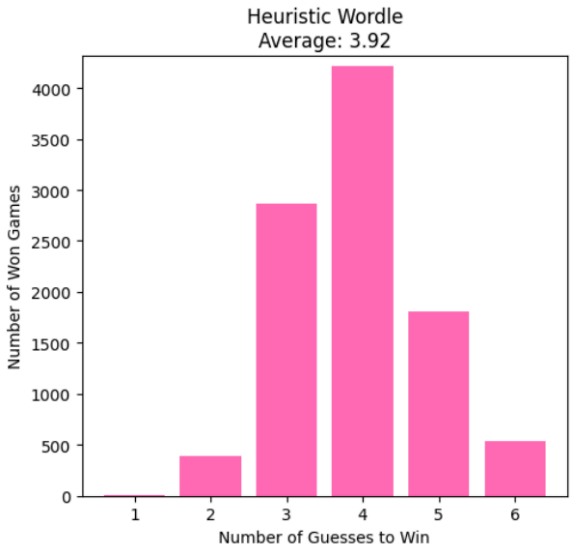




4.3 The Wordle Turtle

The Wordle Turtle, the heuristic solver, is able to beat Wordle an astounding 98.3% of the time. It also achieves our goal of beating the elimination solver in the number of guesses, as it takes on average, 3.92 guesses.





5 Discussion

To note for the following discussion section: There is a lot of cheating in Wordle [1]. For example, “Statistics say about 860 people a day should guess the right word on their first try. Instead, there are upwards of 4,000” [1]. This skews both the win rate and the number of guesses for the global data. 97% of played games are won [8]. In the US it takes approximately 3.92 guesses [9]. As an avid player and non-cheater, over 254 games, Ava has a win rate of 95% and it takes me an average of 4.19 guesses. Ellery has not been logged in when playing and does not have data.

The elimination solver that uses constraint propagation is similar to how a human would play Wordle, with the added benefit of knowing all possible words. It comes close to beating the human stats, again considering cheating. This is expected. It also was able to handily beat the Monte-Carlo tree search solver.

Due to the randomness of the elimination wordle, exact conclusions cannot be made about the words that it misses as they change per run. However in general it misses words that contain less frequent letters like v, x, and z, as well as words with double letters, and words that have ⅚ similar letters to other words (such as daunt, vaunt and taunt). This makes sense because it is randomly picking words from the pruned lists. Less frequent letters are less likely to be in the guessed word which doesn’t allow us to gain information for the constraint propagation.

The Wordle Turtle performs better than the cheaters and better than me in all metrics. The Wordle Turtle also beats the elimination solver as well as the results from the solvers in the related works. This is interesting because of the complexity of reinforcement learning. This shows that solving Wordle as a CSP is efficient, reliable, and accurate.

The missed words in the Wordle Turtle have similar properties to the missed words in the elimination baseline as they both use constraint propagation. However, for a few words, this issue is compounded by the fact that the heuristic actively leans away from using words with uncommon letters. For example, “jaunt” is missed in the Wordle Turtle for the same reason daunt, vaunt, and taunt are missed, but “J” has the lowest frequency out of the 4 starting letters in the first position so it will not be chosen over the other three if it came down to those 4. Because there is no randomness in the Wordle Turtle, the amount of times it misses a word is purely based on how many times it was randomly chosen by the Wordle game over the 10,000 runs. It misses the same words every time it is the answer for the same reasons.

6 Conclusion and Future Works

In conclusion, Yes, CSP techniques are very effective in solving Wordle. In fact, it is more efficient than many other methods. The Wordle Turtle solver beats both our elimination baseline as well as our human baseline (even with cheaters). In the future, based on these results, we could add a way to bypass the elimination to get even more information if the current guesses don’t provide enough. For example, guessing words that might violate constraints would allow us to include more unguessed letters and gather even more information. This might help us guess words like “jaunt”, and also words that have uncommon letters like v, x, and z.

References:

[1] National Post. 2022. Wordle cheating claims spark uproar in online gaming community. Retrieved October 8, 2024, from https://nationalpost.com/news/wordle-cheating-claims.

[2] ProWritingAid. 2023. Does Wordle use plurals? Understanding Wordle's rules. Retrieved October 8, 2024, from https://prowritingaid.com/does-wordle-use-plurals.

[3] YouTube. 2023. Building a Wordle AI - Machine Learning Tutorial [Video]. YouTube. Retrieved October 8, 2024, from https://www.youtube.com/watch?v=UaVJdvT0clo.

[4] D. Quinn. 2022. Solving Wordle Using Monte Carlo Tree Search and Reinforcement Learning. Medium. Retrieved October 8, 2024, from https://medium.com/@devin.p.quinn/solving-wordle-using-monte-carlo-tree-search-reinforcement-725562779c8b.

[5] A. Kho. 2023. Wordle Solver. Andrew Kho Personal Website. Retrieved October 8, 2024, from https://andrewkho.github.io/wordle-solver/.

[6] Mashable. 2022. What is Wordle? The word game explained. Mashable. Retrieved October 8, 2024, from https://mashable.com/article/wordle-word-game-what-is-it-explained.

[7] GeeksforGeeks. 2023. Constraint Satisfaction Problems (CSP) in Artificial Intelligence. GeeksforGeeks. Retrieved October 8, 2024, from https://www.geeksforgeeks.org/constraint-satisfaction-problems-csp-in-artificial-intelligence/.

[8] C. D. Blake. 2022. Over-the-Top Wordle Analysis. C.D. Blake Blog. Retrieved October 8, 2024, from https://cdblake.com/blog/over-the-top-wordle-analysis.

[9] Sportskeeda. 2024. Wordle Statistics: Global Trends and Patterns Across Countries. Sportskeeda. Retrieved October 8, 2024, from https://www.sportskeeda.com/games/wordle-statistics-globe-trends-patterns-countries.

[10] C. Freshman. 2023. A Comprehensive List of Wordle Words. GitHub Gist. Retrieved October 8, 2024, from https://gist.github.com/cfreshman/d5fb56316158a1575898bba1eed3b5da.

[11] C. Freshman. 2023. A Comprehensive List of Possible Wordle Answers. GitHub Gist. Retrieved October 8, 2024, from https://gist.github.com/cfreshman/a7b776506c73284511034e63af1017ee.