Finding Wordle Strategies That Can be Mastered by Humans

Hongyi Zeng
Jinan University, Guangzhou, 511442, China
zengrber@stu2020.jnu.edu.cn

Abstract. Wordle is a popular web game. Players use strategies and hints to figure out the correct answer of a daily puzzle. In general, there are several strategies to solve it efficiently, but which is the best depends on the game's current states. This paper designed 4 individual strategies which is learnable for both computers and human players, and trained an AI based on Q-learning to solve the game automatically with a win rate of over 80%. The decision this AI made is only based on the word list, not including other information (e.g., frequency of appearance on the Internet). Based on the analysis, it can give players advices in various kinds of situations and offer useful combos. By using this AI players, it is feasible to figure out what is the best choice and improve their skills. However, it still cannot solve some rare situations efficiently, which may require more training or a new way of setting the game states.

Keywords: Wordle; Puzzle Games; Reinforcement Learning; Q-learning; Human-computer Interaction.

1. Introduction

Reinforcement learning is a kind of machine learning algorithm that trains the computer to make decisions in different environments. Apart from the traditional machine learning, reinforcement learning is can think and study like a real person. People make decisions based on the current situation and our prediction of each decision's expected reward. Reinforcement learning's basic strategy is almost the same. Markov decision processes (MDPs) have abstracted the learning progress to a cycle that an Agent acts, interacts with the environment and gets reward. From MDPs, if one sets the environment, rewards and optional actions properly, people are able to make machines learn how to make decisions.

Recent year reinforcement has been applied in many games [1]. It gives the computer the ability to play with itself and quickly gain experiences and reach higher scores or defeat human players. The application covers from simple games [2] (e.g., flappy bird and Snake) to very complicated games (e.g., Go) or even RTS (e.g., StarCraft 2). These applications show reinforcement learning's great potential. For any games that needs player to do decisions, reinforcement learning may could learn and master it [3]. Moreover, as the algorithm 'thinks' just like a human being does, people could use the AI as a teacher and improve the skills by understand why the computer makes a certain decision.

In 2021, a web game named Wordle appears on the Internet and gained great popularity. Many people challenge the game and post their solving process on social medias like twitter or Facebook. There are algorithms that solves the game in a very efficient way. Nevertheless, it uses data that people are not able to know and its amount of calculation is too large for people, which means its strategy cannot be mastered by humans [4]. However, Wordle is an interesting game which needs to make decisions in different occasions. After summarizing the strategies that people might use, and the way to find a best actual move for each strategy, an AI could be created to solve Wordle automatically and give learnable advice to people.

Wordle is a 5-letter-words based puzzle game. Players have 6 chances to guess the word which is randomly generated by the computers. After the player guess a 5-letter-word, the game will display the color of each letter as a hint to the player: Green means this letter appears in the correct position, the correct answer has the same letter in this position. Yellow means the correct answer contains this letter, but it is in the wrong position, and gray means that this letter is absent, the correct answer doesn't contain this letter. For example, if the correct answer is FUNNY, if the player guesses BONUS, then the color of each position will be [gray, gray, green, yellow, gray].

In this paper, various strategies and the decision method in different states will be investigated based on Q-learning. Specifically, the usage of the exclude strategy and combo strategy. The rest part of the paper is organized as follows. The Sec. 2 will introduce 4 kinds of strategy which are possible to be used in real Wordle games. The Sec. 3 will set and apply Q-learning algorithm to let the computer find the best choice in different states. The Sec. 4 will analysis the training result and state the remaining issues.

2. Methodology

In normal wordle solving logic, players each time remove every word that cannot be the correct answer and then guess one. Whereas, this strategy sometimes meets a problem: suppose all last 4 letters are correct, and they are: A, T, C, H. Unfortunately, there are still about 7 possible answers (Batch, catch, hatch, latch, match, patch, watch). If the player sticks on the correct letters, one will have a great possibility of losing the whole game. But if one guesses BLOOM, one can immediately know if batch, latch or match is the correct answer. This strategy is defined as 'the Exclude Strategy'. Moreover, there are also combos in this game. Making every simple move the best may not be the best overall strategy. On wordle's official website, many people love using ADIEU as their starting word. Although this is an efficient word, the second guess are more likely to be not-so-good. However, combos (e.g., TRAIN and LOUSE) can have better effect. Using this combo, people can often solve the puzzle in 3 or 4 guesses, but choosing which strategy depends on the current game states. In order to maximize the win rates. This paper designed two algorithms for each strategy and use reinforcement learning to generate a decision list for the computer.

First some basic definitions are demonstrated. If a position's letter has been determined, it is called a correct position. For all the letters that appears to be yellow, they are added into a set named yellow set. Once a letter in the yellow set is fitted in its correct position completely, which means the rest positions don't contain this letter any more, it will be removed from the yellow set. Another list is defined to mark that each position's yellow letters the player guessed, which is called the yellow chart. For those letters which never appears in the word, they are added into a list named absent letters.

Now this paper introduces the abstraction of the normal guessing strategy. From the very beginning, all incorrect words should be removed from the possible word list. The incorrect words meet the following condition:

- It contains incorrect letters in the correct positions.
- It doesn't contain letters in the yellow list.
- It contains letters in the corresponding position's yellow chart.
- It contains letters in absent letters.

These incorrect words could be removed by traversing the whole word list L for 1 time and get a 'possible correct' word list L1. Then, one needs to find a best word in L1 which most efficiently helps to solve the problem. Basically there are two options here, i.e., try to get the most green letters and the most yellow letters. To get the greenest letters, the possibility of a letter appear on the corresponding position needs to be maximized. In order to do this, this paper uses p[i][j] (0<=i<=25, 0<=j<=len(word[i])) to describe the occurrence number of each letter on jth position. And for the ith word in L1, this paper gives them a green score G(i):

$$G(i) = \sum_{j=1}^{len(word[i])} \tan\left(\frac{\pi * p[i][j]}{2len(L1)+1}\right) \quad (1)$$

It is used to evaluate each word in L1 because finding a new green letter is the most important mission for this strategy. The p[i][j]/len(L1) shows the probability that the ith letter to be the correct letter in jth position. To make this algorithm learnable and increase its effect, when a letter's probability is higher, it should weigh more, since the goal is to find new green letters. If a letter's correct probability is nearly 100%, during this guess, words which has other letters in this position should not be considered. Therefore, this paper chooses the tangent function to describe each word's weight(score). When p[i][j]/len(L1) is nearly 1, its contribution to G(i) will be extremely large that

other positions are almost inconsequential. After calculation of each word's green score, a word in the top 4 high score words will be picked randomly as the algorithm's final guess.

To get the most yellow letters, the possibility of a letter appear in the answer in any position needs to be maximized. Notice that the position is not important anymore, so this paper uses p[i](0<=i<=25) to describe the occurrence number of each letter. As for the ith word in L1, this paper gives them a yellow score Y(i):

$$Y(i) = \sum_{j=1}^{len(word(i))} \begin{cases} \tan\left(\frac{\pi * p[int(word[j] - 'A')]}{2len(L1)+1}\right), j^{th}\ letter\ firstly\ appears\ in\ word[i] \\ 0,\ there\ are\ same\ letters\ before\ j^{th}\ letter \end{cases} \quad (2)$$

Each different letter only needs to be count once because the goal is not finding green letters but yellow letters. the most important thing is if the letter appears but not the position, so words with repeated letters are not efficient. Its score should be reduced. After calculation each word's green score, this paper randomly picks a word in the top 4 high score words.

Now this paper will introduce how to do the exclude strategy [5]. The exclude strategy's goal is to quickly find yellow letters by enlarging the word list. In other words, excluding the limit of green and yellow letters will greatly increase the guessable word list, although some words are impossible to be the correct answer, it may offer more information than normal guesses. One notices that while players are using the exclude strategy, they do not need to focus on the green letters. This is because when players are using the exclude strategy, what they want is more letter's information, i.e., whether it is green or yellow or even absent. Considering the position will sometimes bring us to words with repeat letters, which is what players don't like to see, this paper uses Y function instead of G for the word list's processing. Clearly, players don't need all the letters appear in the previous guesses. Therefore, the best situation is to create a word list removing all these letters. However, this may create an empty word list, so when the list is empty, this paper removes all the absent and yellow letters in the previous guesses. If this still creates an empty list, this paper then only removes absent letters in the previous guesses. At this time, the list must be non-empty since at least the correct answer will be in the word list. After creating a word list L2 using the method just mentioned and calculate the yellow score for each words using Y function. Finally, this paper randomly picks a word in the top 4 high score words.

The single guess part is finished, now the paper will introduce the combo part [6]. As a combo occupies 2 chances, combo strategy's main target is to use a pair of words to quickly exclude or ensure as much letters as possible. Hence, making a guess with same letters in the word doesn't make sense. At the very beginning, the paper removes all the word with same letters in it and make a new word list named unique list, and remove all words that contains the absent letters. For each word in the processed unique list, the paper defines its score:

$$score[i] = \sum_{j=1}^{len(word[i])} p[int(word[j] - 'A')] \quad (3)$$

It should be noted that for combos, different letters do not need different weights given by the tangent function because all players want is to find a best pair with 10 different letters. Subsequently, this study sorts the words descending by its score and find 4 pair with the highest score which doesn't contain repeat letters. Finally, a pair is randomly selected and 2 guesses are made accordingly. However, there is a problem that a pair with 10 different letters may not exist at all. This situation shows that the combo strategy is not working. If player cannot make a pair like this, clearly making a single guess would be a better choice, so one just returns a random guess to make this strategy very bad. This will make the computer less likely to choose it after the deep learning.

Now all 4 strategies have been introduced: Normal guess for green letters, normal guess for yellow letters and the exclude strategy. The next step is to train a model to guide the computer to make the best decision [7]. This problem can be transformed into a reinforcement learning problem. States are described in a pair (m, n), which means there are total m green words and n yellow words. The whole wordle game is the environment, and policies are those 3 strategies. Set the rewards as following:

- 5 points for each green letter
- 2 points for each yellow letter

- 50 points for victory
- -20 points for failure

Now, a basic reinforcement learning model is created. This study trained the model using Q-learning with TD updates. The model's goal is to maximize the expected rewards: $max_\pi E[\sum_{t=0}^{H} \gamma^t R(S_t, A_t, S_{t+1})|\pi]$. Here, π(a|s) means taking an action in s state. R(s'|s, a) means a action's reward that causes s state change to s'. Q-learning is a net which has an auxiliary list that helps to make decision in different states, named Q-List. In each iteration the computer uses the Q-List and epsilon-greedy algorithm to choose actions and Update the Q-List with the feedbacks.

Epsilon-greedy algorithm is an algorithm that sets an $\varepsilon \in (0.0, 1.0)$. In each iteration, one needs to generate a random number P. If $P > \varepsilon$, and then choose the action which has the largest value in Q-List. Otherwise randomly choose an action. Normal epsilon-greedy algorithm needs to exploit knowledge, but in our case, a Q-List already exists, so it does not need to exploit anymore. TD updates can be described as following steps:

- Calculate the expectation of the reward of s state V(s).
- Let R be the reward after taking the action, calculate the TD error: $\delta_t = R + \gamma V(s') - V(s)$.
- Update the Q-List: $Q(s,a) \leftarrow Q(s,a) + \alpha\delta_t$ [8]

Set the learning rate α=0.02, the discount factor γ=0.05. And the boundary value of the epsilon-greedy algorithm ε=0.02. The training rounds is set to 10,000 times. This paper values the winning rate by calculating recent 1000 play's win rate.

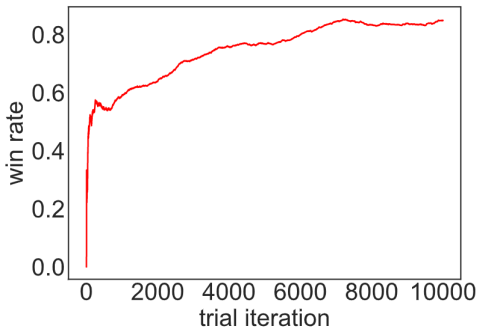

Fig 1. Winning rates of trial iterations.

3. Results & Discussion

As shown in Fig.1, the winning rate quickly raised to 80%. The maximum winning rate of recent 1000 plays is up tp 87.8%. Seen from Fig. 2, normal guess for green letters is the best strategy for most of the time. But if there are only few yellow letters and no green letters, then finding a combo that exclude letters quickly will be a better strategy [9, 10]. When there are 2 green letters and no yellow letters, the computer thinks the best strategy is using the exclude strategy. It makes sense since in normal plays, (2, 0) state mostly happens after 2 or 3 guesses. In this situation, combo strategy will face high risk and may not work at all. At most time players still have chances to make wrong guesses. So, exclude strategy can be a good option. This result could be used as a reference to humans. In the guessing algorithm, the letters with higher probability will have a higher weight, which means players are able to imitate it. Because one can know which letter has a higher probability to appear in the words, and only needs to find out a letter which is the most likely to appear. Combined with the previous guesses and current state, it is available to figure out a guess which its method is approximately the same as the computer's algorithm. Besides, computer can also find combos in different situations and in the very beginning, where player can simply memorize them to improve their winning rates.

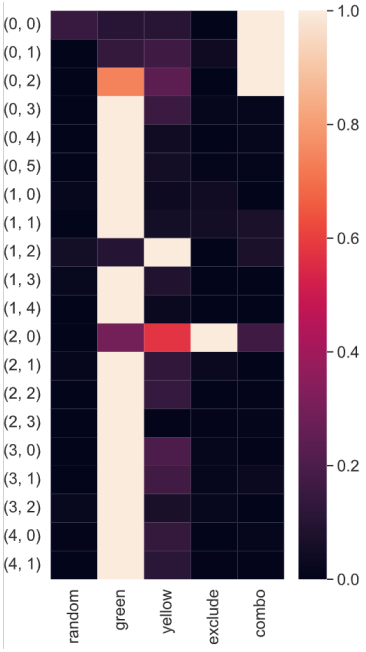

Fig 2. Prefer Strategies in different states.

Loading the Q-list into the test environment and automatically plays 10,000 times. The computer got a total winning rate of 86.72%. This data is collected by using wordle in New York Times' word list, which has 12,972 words. In addition, if players change the word list to a smaller one, then the winning rate will become over 97%, which is very high.

After testing, the situation mentioned in the very beginning hasn't been solved properly. This is because (4,0) state usually happens after 4 or 5 guesses. In the next guess there are a great possibility to end the game. Meanwhile, (4, 0) state itself is very rare in the game play. Hence, there aren't enough training for this state. In order to improve the situation, one might try to redefine the state to (Y, G, R) which Y means the number of yellow letters, G means the number of green letters, and R means the remaining chances. However, this method requires gigantic trainings and takes a lot of time, and its argument are more complex to set. Further study will continue focus on this problem and try to find a better learnable playing strategy.

4. Conclusion

In summary, this paper investigates learnable Wordle game strategies based on deep learning. To be specific, this paper designs 4 strategies: normal guess (maximizing green letters), normal guess (maximizing yellow letters), exclude strategy and combo strategy, and trained the computer to figure out which strategy is the best one in different states. According to the analysis, at most time, normal guesses (maximizing the green letter) is the best strategy; combo strategies work better in the beginning of the game; the exclude strategy is used when there are still many chances left and there are already some green letters. However, the extreme rare situations are still not solved. Further research will try to redefine the states and increase the trial iterations to challenge higher winning rates. This research uses reinforcement learning instead of traditional guessing method. It has 2 main advantages: learnable and higher winning rate than other reinforcement algorithm. Regarding to some problems, one might face in the real game play is still not solved, hope our research can provide a method for the wordle challengers and give other researchers a little bit inspiration. Overall, these results offer a guideline of how to solve the Wordle game in a few-knowledge and logic way for both people and computers.

References

[1] Heinrich, J., Silver, D.: Deep reinforcement learning from self-play in imperfect-information games. arXiv preprint arXiv:1603.01121 (2016).

[2] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013).

[3] Brown, N., Bakhtin, A., Lerer, A., Gong, Q.: Combining deep reinforcement learning and search for imperfect-information games. Advances in Neural Information Processing Systems, 33, 17057-17069 (2020).

[4] Rikters, M., Reinsone, S.: How Masterly Are People at Playing with Their Vocabulary? Analysis of the Wordle Game for Latvian. arXiv preprint arXiv:2210.01508 (2022).

[5] A Wordle Hack, 2022, Retrieved from: https://thamara.blog/a-wordle-hack-aa83912cd979.

[6] de Silva, N.: Selecting Seed Words for Wordle using Character Statistics. arXiv preprint arXiv: 2202. 03457 (2022).

[7] Anderson, B. J., Meyer, J. G.: Finding the optimal human strategy for wordle using maximum correct letter probabilities and reinforcement learning. arXiv preprint arXiv:2202.00557 (2022).

[8] Omidshafiei, S., Papadimitriou, C., Piliouras, G., et al.: α-rank: multi-agent evaluation by evolution. Scientific reports, 9(1), 1-29 (2019).

[9] Liu, C. L.: Using Wordle for Learning to Design and Compare Strategies. arXiv preprint arXiv: 2205. 11225 (2022).

[10] Bonthron, M.: Rank One Approximation as a Strategy for Wordle. arXiv preprint arXiv:2204.06324 (2022).