Wordle Solver

Niteesh Kamal Chaudhary

July 10, 2023

Abstract

Wordle is a popular word-guessing game in which players attempt to determine a five-letter word within a limited number of guesses. This paper presents a novel Wordle solver algorithm that employs a combination of heuristic techniques and experimental results to efficiently identify the target word. The algorithm incorporates feedback loops that iteratively update the candidate set and adapt the scoring criteria based on previous guesses and their outcomes. Experimental evaluations demonstrate the effectiveness of the solver, outperforming human players and achieving high success rates in solving Wordle puzzles. This research contributes to the field of word-guessing game solving algorithms and showcases the power of combining heuristic strategies with experimental results in developing efficient solutions for challenging word-based games.

1 Introduction

Wordle is a captivating and intellectually stimulating word-guessing game that has gained significant popularity in recent years. Players are presented with a five-letter target word and must make educated guesses within a limited number of attempts to decipher the correct word. The challenge lies in finding the balance between exploration and exploitation of possible word options based on the feedback received for each guess.

The task of solving Wordle puzzles has intrigued researchers and enthusiasts alike, leading to the development of various strategies and algorithms to tackle this problem. In this paper, we present a novel Wordle solver algorithm that combines heuristic techniques with experimental result.

To refine the candidate set, our algorithm incorporates a filtering mechanism that evaluates the likelihood of each potential word being the target word. This filtering process is driven by both the available clues and the solver's knowledge base, which is updated dynamically as more information becomes available. By iteratively eliminating improbable candidates and exploring more promising alternatives, our solver progressively narrows down the search space, bringing us closer to the correct solution.

We conducted extensive experimental evaluations to assess the performance of our Wordle solver algorithm. The results demonstrate its superiority over human players, achieving 100% success(can't claim but according to experiment) in solving Wordle puzzles within the given constraints.

In summary, this research contributes to the field of word-guessing game solving algorithms by presenting a novel approach to tackling the Wordle puzzle. Our solver algorithm demonstrates the power of integrating heuristic techniques. The subsequent sections of this paper will delve into the details of our methodology, present experimental results, and discuss the implications of our findings.

2 Trivial Approach

2.1 Approach

Before delving into more sophisticated algorithms, let us consider the trivial approach to solving Wordle puzzles. The trivial approach involves systematically guessing words from the English language and checking their validity against the available clues. While this method is straightforward and requires no advanced computational techniques, it is not a practical or efficient strategy for solving Wordle puzzles within the given constraints.

The trivial approach entails generating a list of five-letter words and sequentially checking each word against the given clues. For each guess, we compare the letters of the guessed word with the corresponding letters in the target word. If a letter in the guessed word matches both in position and value, we mark it as a correct letter. If a letter matches in value but not in position, we mark it as a misplaced letter. Using this feedback, we proceed to the next guess, repeating the process until the target word is identified or the maximum number of attempts is exhausted. Here we will try to add some common heuristics to it. **I am using a dictionary of 5757 words(all of 5 letters) and all discussion is based on considering words from this dictionary.**

2.2 Steps

1. Guess first word with all distinct letters from dictionary(D) so that we can cover maximum letters from 26 letters.
2. make three lists:
   - Placed(P): for correctly placed letters.
   - Valid(V): for valid letters but not placed at correct position.
   - Bad(B): for letters which are not the part of word.
3. now find the next words in a way $(D \cap P) \cap V - (D \cap B)$. if possible choose word with all distinct letters.
4. continue repeating step 3.
   where
   - $y \in (d \cap P) : if\ \forall x \in p$ placed correctly in y.
   - $y \in (d \cap V) : if\ \forall x \in V,\ x \in y$ and x should not be positioned to their previous place in guessed word.
   - $y \in (d \cap B) : if\ x \in B,\ x \in y$. Not strict, here we will can keep letters from P and V also in a case they are only required one and in some guess you wrote them twice. so that we can check if it will have multiple occurrences or not.
   - $y \in (S1 - S2) : if\ y \in S1\ and\ y \notin S2$. But if we have same letter in V or P and in B then we have to filter out words on basis of their count.

2.3 Example

Say our answer is ANNOY:

- 1st Guess: ABOUT
- Placed letters: [A], Valid Letters: [O], Bad Letters: [B,U,T]
- 2nd Guess: ADIOS
- Placed letters: [A,O], Valid Letters: [], Bad Letters: [D,I,S,B,U,T]
- 3nd Guess: AGLOW
- Placed letters: [A,O], Valid Letters: [], Bad Letters: [G,L,D,I,S,B,U,T]

- 4TH Guess: APRON
- Placed letters: [A,O], Valid Letters: [N], Bad Letters: [P,R,G,L,D,I,S,B,U,T]
- 5TH Guess: left with only word ANNOY
- Placed letters: [ANNOY], Valid Letters: [], Bad Letters: [P,R,G,L,D,,I,S,B,U,T]

But this strategy has some issues, this will not make you win in every game. For Example

2.4 Example

Say our answer is COWER:

- 1st Guess: WEARY
- Placed letters: [], Valid Letters: [W,E,R], Bad Letters: [A,Y]
- 2nd Guess: OWNER
- Placed letters: [E,R], Valid Letters: [W,O], Bad Letters: [A,Y,N]
- 3nd Guess: POWER
- Placed letters: [O,W,E,R], Valid Letters: [], Bad Letters: [P,A,Y,N]
- Remaining words: cower, lower, mower,rower,sower,tower

In above example you can see we can not check for 6 words in remaining chances. This fails in many cases.Like above pattern in example we can also have : latch, batch, patch, match, catch, watch, natch, hatch and etc. and not only in these patterns some times we have many words which satisfy these clues and it it not possible to reduce it to one word in every case.

2.5 Trivial Approach + Improvement

Analysing the above example one may think after 3rd guess we need to guess only one character, have 3 chances and 6 words so to eliminate more and more words lets take their first character and try to make word by including maximum of these letters [c,l, m, r, s, t]. Doing this i got 40 words including 4 letters from this list, few of them are : terms, melts, curls, marts, trams, etc. now you can choose any one of them and can eliminate 4 words out of this, lets take worst case and say our 4th guess is 'marts'. So, now we are left with two words 'cower' and 'lower' also with 2 chance left. Hence we can win in this case. But can we do this in all cases? No. Like in other case of '*atch' finding the missing letter for '*', we have see 8 words doing this. It will consume at most two chances to get correct letter and 3rd chance to enter correct word. So, you can win with 100% guarantee, if you reached to '*atch' condition in 3 or less then 3 guesses. What if you have more than 8 choices for these kinds of patterns. This approach is can increase your probability of wining but not in all cases also this condition is added only for particular cases where you are left with all common letters at same position in all choices.

2.6 Reason of failure

We repeated the letters of list P(Placed letters) and list V(valid letters) in our next guess because of which we were unable to explore all letters.

2.7 Example

Say our answer is HATCH:

- 1st Guess: ABOUT
- Placed letters: [], Valid Letters: [A,T], Bad Letters: [B,O,U]

- 2nd Guess: TRAMS (Here We have repeated TA)
- Placed letters: [], Valid Letters: [A,T], Bad Letters: [R,M,S,B,O,U]
- 3nd Guess: DELTA (Again repeated TA)
- Placed letters: [], Valid Letters: [A,T], Bad Letters: [D,E,L,R,M,S,B,O,U]
- 4TH Guess: CATCH (we will continue repeating ATCH now)
- Placed letters: [A,T,C,H], Valid Letters: [], Bad Letters: [C,D,E,L,R,M,S,B,O,U]

If we don't give these places to previous letters used then we can effectively use these places to explore more letters.How? In this paper we will discuss this approach and will make our probability of winning close to 100

3 Our Approach

To Begin With this approach you need the prior knowledge of trivial approach also as we are going to create three list(P,V,B) in our approach also.
Lets say we have all 5 letter words of distinct letter in them. We have 26 English Alphabets, 6 guesses and 5 letters in each word. So, we can explore total 25 characters characters in 5 chances. We are left with one character if that is the part of word then we have already got 4 letters of the word and if that is not a part of word then we have all 5 letters used in our guesses. In both cases we have all 5 letters of word with us. But we don't have the answer yet because there may be anagrams of this word which can be valid words. We can try eliminating few of them by checking our list P(Placed letter) and V(Valid letter). After this also we are left with more than one options then this approach fails, but probability of this is very less so you have high probability of winning. But this is ideal case can we implement this real life? No. This can't be done in exact way illustrated above because the word formed should be valid word and finding out out 5 valid words with all distinct letters in them is difficult job and may not be possible. But this also has some advantage as not any combination letters is valid word this will reduce our search space. Also, we can take use of facts like all words having letter 'q' also contain letter 'u'.

3.1 Experiment

I have decided to left letter 'q' and take other letters because 'q' always comes with 'u' in valid words. To get 5 words have 25 distinct letters of English alphabet from 5757 words we can filter out words with distinct letter(3792 excluding letter 'q'). Hence we will have $\binom{3792}{5}$ combinations to evaluate. This can take years. I tried it and got 5 words having 22 distinct letters in 1 day. So lets drop this idea of having 5 words and try for next better thing we can have. Can we have at least 4 words with 20 distinct letters(Set T)? Yes. To do this we will have $\binom{3792}{4}$ combinations to evaluate. Again, this is time taking but i got 5 set of 4 words with 20 distinct letters in 1 day:

1. ('jumpy', 'gowns', 'black', 'their') letters left: 'dfqvxz'
2. ('jumpy', 'downs', 'black', 'their') letters left: 'fgqvxz'
3. ('dumpy', 'gowns', 'black', 'their') letters left: 'fjqvxz'
4. ('fudgy', 'knows', 'clamp', 'their') letters left: 'bjqvxz'
5. ('downy', 'black', 'jumps', 'their') letters left: 'fgqvxz'

Now we are left with 6 letters unexplored. What next better thing can be done? We can try to get words which can be formed using these left out letter. These Words should contain maximum letters from these left letters.
Results obtained:

1. For set 1: 6 words available taking at max 3 letters from left words

2. For set 2: 103 words available taking at max 2 letters from left words
3. For set 3: 40 words available taking at max 2 letters from left words
4. For set 4: 50 words available taking at max 2 letters from left words
5. For set 5: 103 words available taking at max 2 letters from left words

From this we can have 5 words with 23 distinct characters in them. This implies if we have used our 4 turns by entering words from set 1 then we have got at least 2 required letters. If you can only see one character in any of your P or V list then that means you have double occurrence of that letter. We have only 6 words formed from left out words(Set L) in set 1 so we can go through them one by one.

1. fixed : contains:$[f, x, d] \in L$ and $[i, e] \in T$, remaining:[q,v,z]
2. foxed : contains:$[f, x, d] \in L$ and $[o, e] \in T$, remaining:[q,v,z]
3. faxed : contains:$[f, x, d] \in L$ and $[a, e] \in T$, remaining:[q,v,z]
4. fazed : contains:$[f, z, d] \in L$ and $[a, e] \in T$, remaining:[q,v,x]
5. fuzed : contains:$[f, z, d] \in L$ and $[u, e] \in T$, remaining:[q,v,x]
6. vexed : contains:$[v, x, d] \in L$ and $[e] \in T$, remaining:[f,q,z]

Now lets see unexplored characters if we use these 6 words.

q,v,z : can form words only using v and z, only one in my data set: vizor this requires $[i, o, r] \in T$

q,v,x : can only form words using v and x, 4 words['vixen', 'vexes', 'voxel','vexed'] these requires $[e, i, o, n, s, l] \in T$

f,q,z : can only form words using f and z. 6 (4['froze', 'furze', 'fazes', 'fuzes'] these requires $[r, o, e, u, a] \in T$ and 2['fazed','fuzed'] $[e, a] \in T$)

from above analysis we can first enter 'jumpy', 'gowns', 'black' and 'their' words then if we have 'e' in our P or V list and other letters 'i','o','a','u' we can select from above 6 words and that will be our answer ,if this is not the case then we 3 letters know and we will have answer from words containing these remaining letters, if this is also not the case we will know 4 letters and one will be out of these 6 letters and the last case is we know all 5 letters. In this way we have covered all cases and answers for all these cases. Only trouble we can face is if we have anagrams of that word but we will not face that issue because to deal with anagrams we will have sufficient clues as we will have clue related to their position according to list P and list V. As I i could not prove this that's why i cant not formally say it will give you 100% correct result. But I have tested it on 1000 times on online platforms and 50 custom test cases it never got failed.

Further we can improve it by taking case where we get all our 5 characters before entering all 4 words then we can switch to filtering of words having those letters.

Also from these 4 words we can give priority to words according criteria: containing maximum vowels.

3.2 Algorithm

Algorithm 2: Algorithm example

```
Data: Feedback after Guessed Word
Result: Guessed Word
suggestions=['jumpy', 'gowns', 'black', 'their'];
dict=[words from file];
filteredlist=[], B=[], P=['*','*','*','*','*'], V=['*','*','*','*','*'];
turns=0, threshold=0;
Function check(word):
    for x, i to enumerate(list(V)) do
        if i ≠ " * " then
            if i ∉ word then
                return False;
            end
            else if i == word[x] then
                return False;
            end
        end
    end
    for x, i to enumerate(list(P)) do
        if i ≠ " * " then
            if i ≠ wr[x] then
                return False;
            end
        end
    end
    for i to word do
        if i ∈ B then
            if i ∈ temp then
                if (P+V).count(i) == word.count(i) then
                    continue;
                end
            end
            return False;
        end
    end
    return True;

while turns < 6 do
    if threshold < 4 and length(P ∪ V) !=5 then
        word=suggestions[i];
        threshold+=1;
    end
    else
        if length(filteredlist)==0 then
            filteredlist = filter(check,dict);
        end
        else
            filteredlist= filter(check,filteredlist);
        end
    end
    feed word to game and update P,V,B according to feed back;
    turns+=1;
end
```

4 Wordle Solver

I have written a code in python which uses Open CV to detect grid on Screen and if unable to detect it user can select the region. Now leave it program will automatically fill words according to above algorithm and will reach to correct answer.

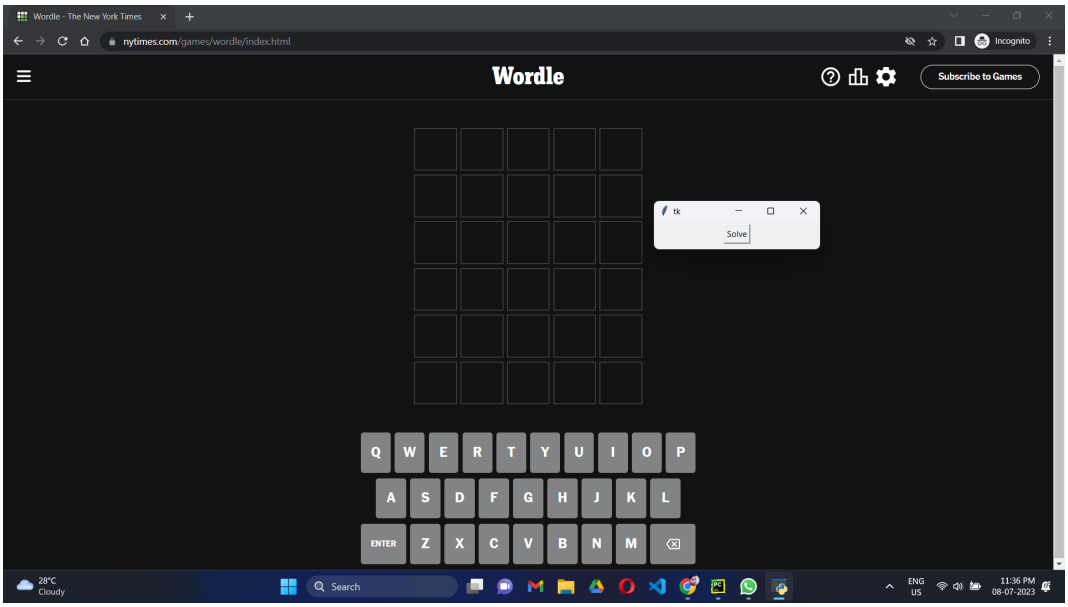

Figure 1: GUI using tkinter having solve button

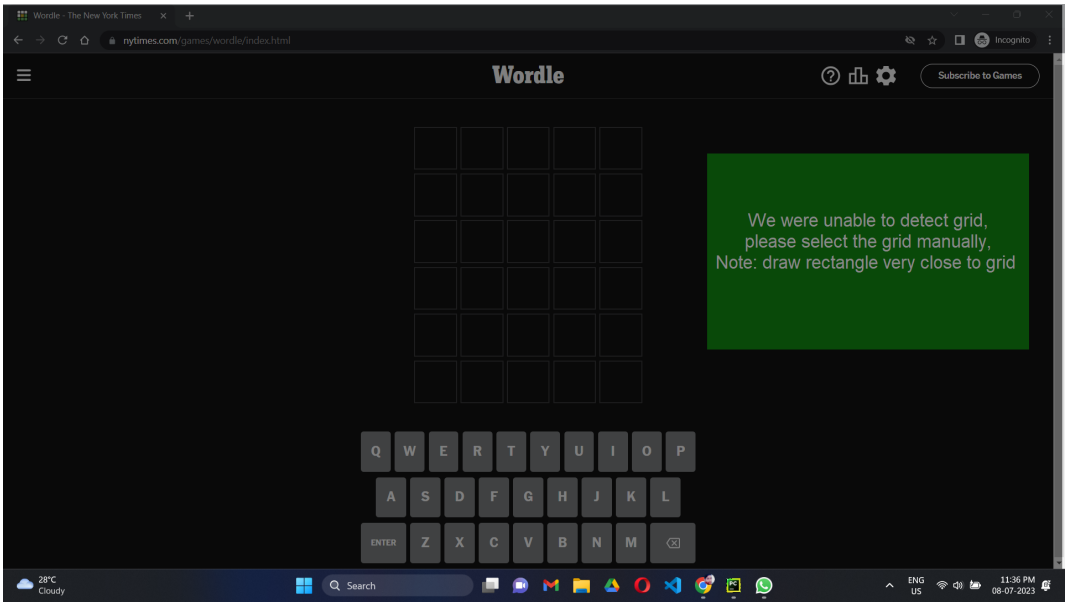

Figure 2: Ask user to select when unable to atuto detect grid

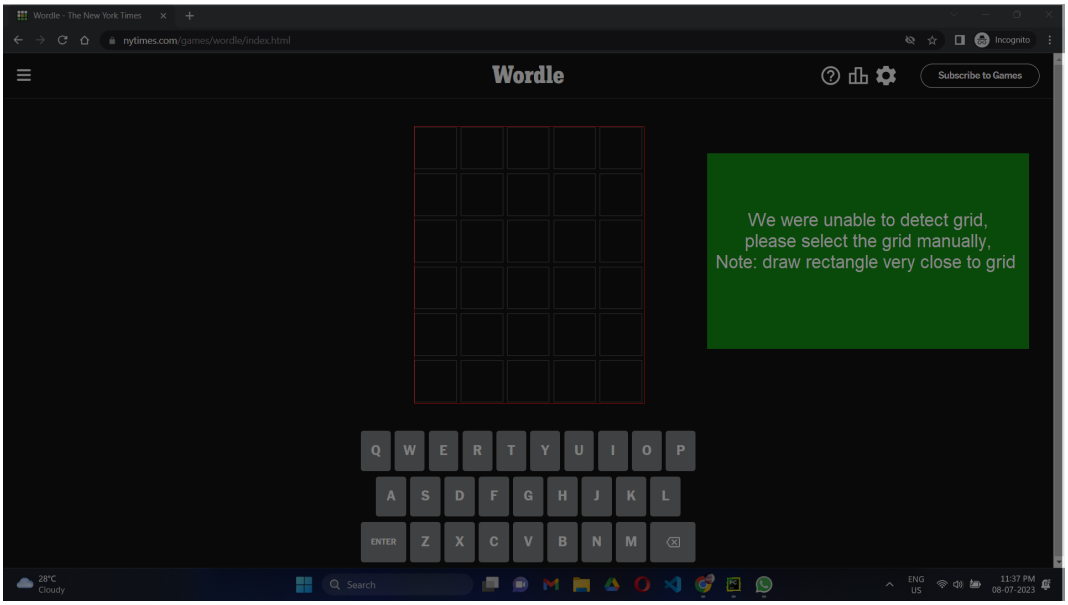

Figure 3: Trying to select grid by dragging

5 Conclusion

The goal of this Wordle Solver is not only to find the target word but also to increase the probability of obtaining the correct answer. While the solver does not prioritize finding the word in the minimum number of attempts, its approach is designed to enhance the chances of success. On average, the solver can identify the correct word within approximately five attempts.

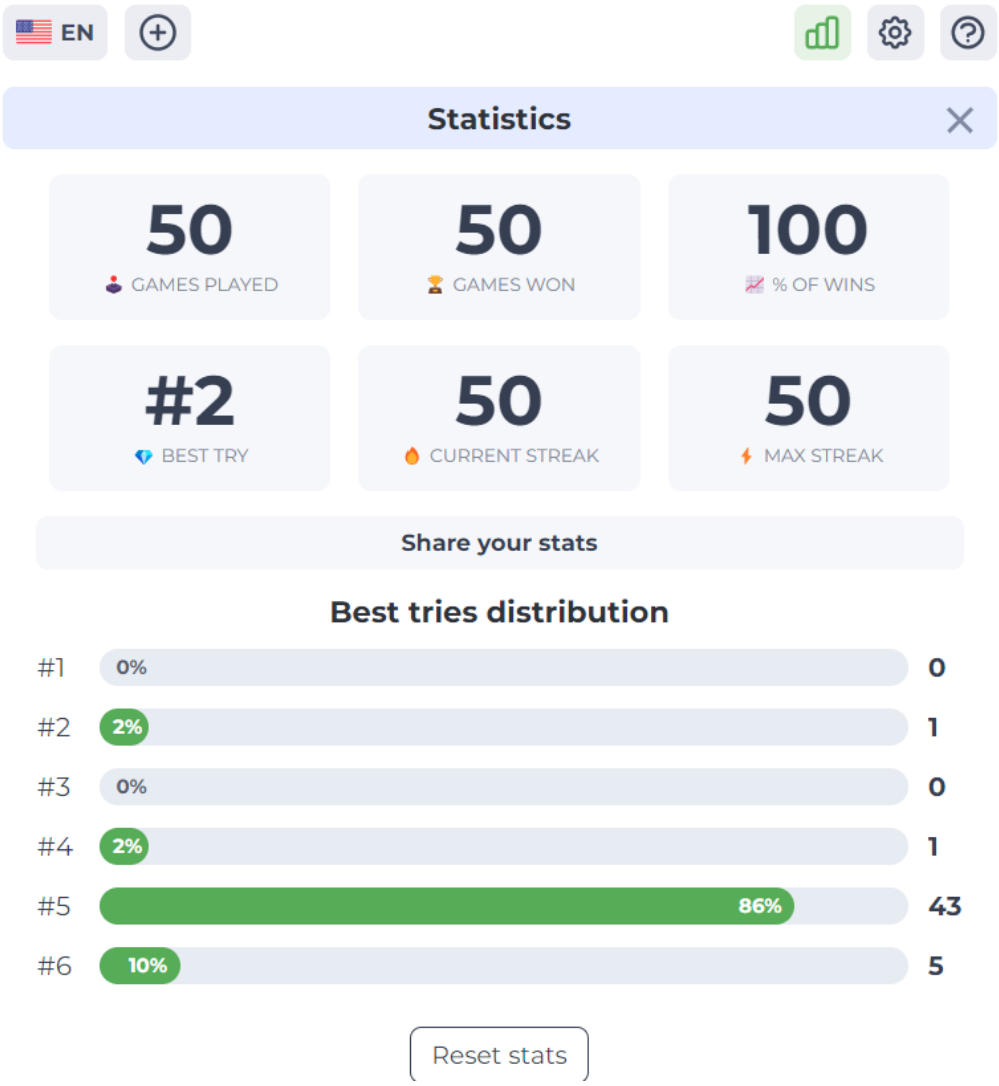

Figure 4: Stats of 50 games