Solving Wordle using Information Theory

ITC Course Project
Vatsala Nema


WORDLE
S T O O D
D O T E S
T O A D S
T O D A Y
TODAY

WORDLE

1. Every letter that is not present in the solution will have its background turn gray.
2. The background of letters that are present in the solution but are misplaced will become amber.
3. Letters that are both present in the solution and placed in the correct position will get a green background.

   This rule overrides the previous rules, and therefore duplicates of a correctly placed letter in the input will not receive an amber background unless the letter is also repeated in the solution.

Problem Statement

Predicting the best possible starting word for the game **Wordle** based on the Entropy of the word and then solving it in minimum possible trials.

Precomputation

Perform a green, and amber pass over the corpus set and determine final patterns of all solutions with all possible words.

```
def greenPass(word1, word2, used1, used2):
    result = 0
    for i in range(5):
        if word1[i] == word2[i]:
            result += 2 * 3**(4-i)
            used1[i] = used2[i] = True
    return result
```

```
def amberPass(word1, word2, used1, used2):
    result = 0
    for (i,c1) in enumerate(word1):
        for (j,c2) in enumerate(word2):
            if c1 == c2 and not (used1[i] or used2[j]):
                result += 3**(4-j)
                used1[i] = used2[j] = True
                break
    return result
```

```
def pattern(word1, word2):
    used1 = [False for _ in range(5)]
    used2 = [False for _ in range(5)]
    return ( greenPass(word1, word2, used1, used2)
           + amberPass(word1, word2, used1, used2) )
```

Entropy based Approach

- Ranking of possible words from the corpus based on Entropy.
- Reduction in sample space using conditional entropy.
- Testing words incompatible with the pattern
- After every trial of 5 lettered word - words that are incompatible with the pattern should be eliminated from it, or should not make it into further attempts to reach the solution
- Identify each of the 3^5 possible patterns by its ternary representation, where 0 identifies gray, 1 identifies amber and 2 identifies green.

$$\mathcal{H}(X_w) = -\sum_{i=0}^{3^5-1} p_{X_w}(i) \log_2 p_{X_w}(i)$$

Naive Algorithm

```
def play(solution, allowed=words, starter='tares', mode='naive'):
    remaining_words = np.array(allowed, dtype = 'str')
    next = starter
    for attempt in range(9):
        pattern = get_pattern(next, solution)
        if pattern == 242:
            break   # found the solution!
        remaining_patterns = get_pattern_matrix(next, remaining_words)
        remaining_words = remaining_words[remaining_patterns.flatten() ==
    pattern]
        i, next = make_guess(remaining_words, mode)
        remaining_words = np.delete(remaining_words, i)
    return attempt
```

Approach: Greedy Algorithm

Next step: Leverage commonly occurring words in the priority list even if they have a lower entropy.

The greedy algorithm will always choose the word that reaches the following maximum:

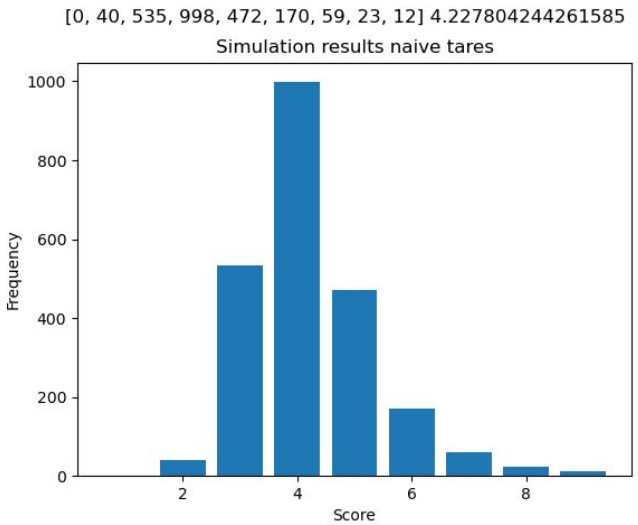
$$\max_{w \in G} \{ g(r_w) \cdot \mathcal{H}(X_w) \}$$

where g : R → (0, 1] is a con[illegible] on that we must define.

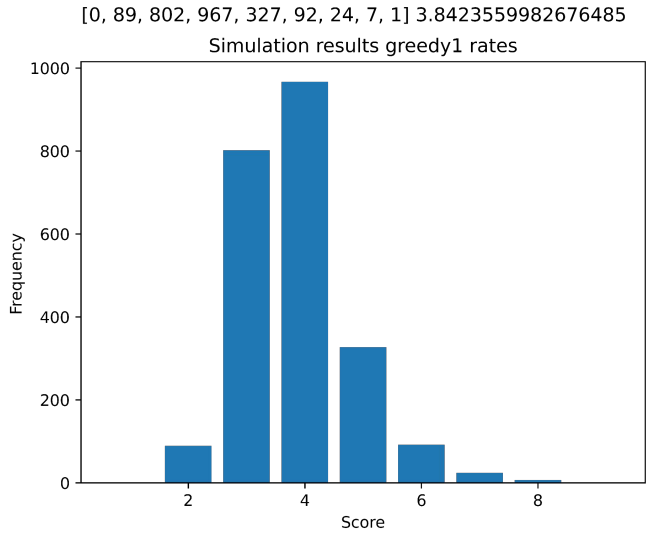
```
@cache
def make_guess(wordlist, mode):
    patterns = get_pattern_matrix(wordlist, wordlist)
    l = len(wordlist)
    maxe = best = -1
    for i,w in enumerate(patterns):
        # List of distinct patterns and their counts
        ps, counts = np.unique(w, return_counts=True)
        e = - np.dot(counts/l, np.log2(counts/l))
        if mode == 'greedy1':
            e *= get_word_priority(wordlist[i])
        if e > maxe:
            maxe, best = e, i
    return best, wordlist[best]
```

PO:Find the solution to any Wordle puzzle in as few attempts as possible using the words from the list of allowed words.

Score: 4.224



Time Taken: 20 mins, 17 seconds

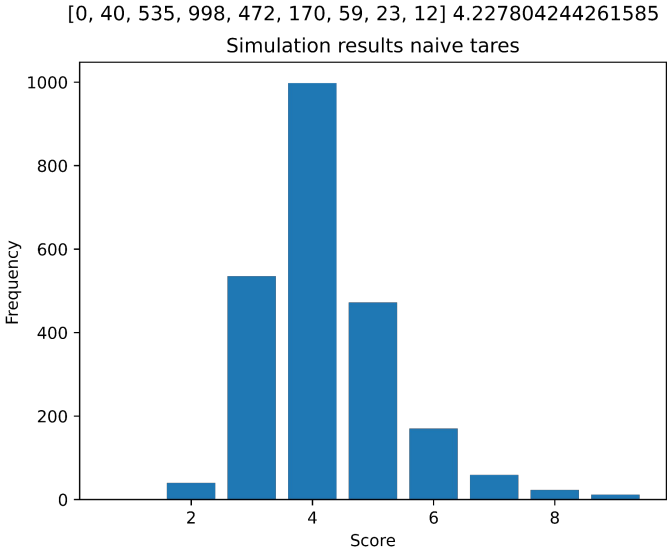


Time Taken: 6 mins, 42

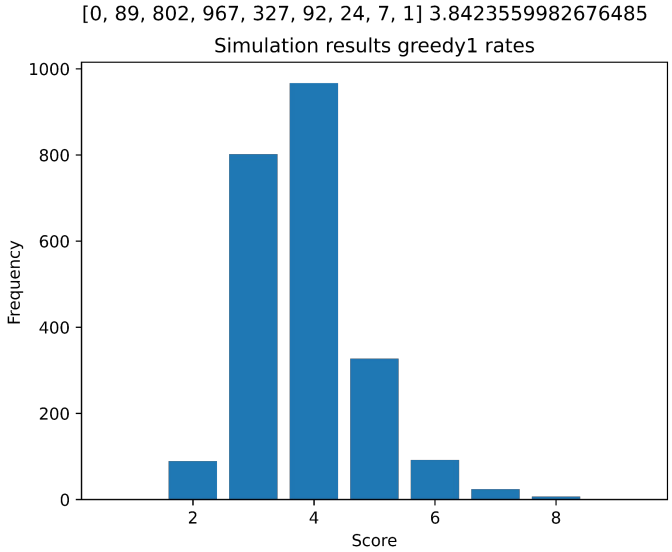
P1:Find the solution to any Wordle puzzle in as few attempts as possible using *only words from the list of solutions.*

Score: 3.84

$$\mathcal{H}(X) - \mathcal{H}(Y) = \log_2 |G| - \log_2 |S| = \log_2 12972 - \log_2 2309 = 2.49 \text{ (bits)}$$



Time Taken: 19 seconds



Time taken: 18 seconds

Future Directions

1. Optimality Analysis

   Come up with a robust and time efficient algorithm
2. Using decision tree/genetic algorithms to further increase
3. Controlling for a more frequent starter word

RESULTS

```
(myenv) C:\Users\VATSALA NEMA\Documents\SEM 8\Info theory\CustomWordleAlgs-main\CustomWordleAlgs-main>C:/Apps/Conda/envs/my
env/python.exe "c:/Users/VATSALA NEMA/Documents/SEM 8/Info theory/CustomWordleAlgs-main/CustomWordleAlgs-main/InformationTh
eoryAlgorithm/Simulatorv2.py"
Choose mode: [naive] or [greedy1] or [greedy2]
greedy2
Choose starter word or press [enter]
rates
Using [ rates ] as starter for [ greedy2 ] simulation
100%|| 2309/2309 [00:05<00:00, 390.36it/s]
Results = [0, 118, 976, 999, 173, 34, 6, 3, 0]
Score = 3.59246427024686
Stop simulations? y/n
y

(myenv) C:\Users\VATSALA NEMA\Documents\SEM 8\Info theory\CustomWordleAlgs-main\CustomWordleAlgs-main>C:/Apps/Conda/envs/my
env/python.exe "c:/Users/VATSALA NEMA/Documents/SEM 8/Info theory/CustomWordleAlgs-main/CustomWordleAlgs-main/InformationTh
eoryAlgorithm/Simulatorv3.py"
Choose mode: ['naive', 'greedy1', 'greedy2', 'greedy3', 'greedy4']
naive
Choose starter word or press [enter]

Using [ tares ] as starter for [ naive ] simulation
100%|| 2309/2309 [00:19<00:00, 115.45it/s]
Results = [0, 40, 535, 998, 472, 170, 59, 23, 12]
Score = 4.227804244261585
Stop simulations? y/n
n
Choose mode: ['naive', 'greedy1', 'greedy2', 'greedy3', 'greedy4']
greedy1
Choose starter word or press [enter]

Using [ rates ] as starter for [ greedy1 ] simulation
100%|| 2309/2309 [00:14<00:00, 162.00it/s]
Results = [0, 89, 802, 967, 327, 92, 24, 7, 1]
Score = 3.8423559982676485
Stop simulations? y/n
```

```
Using [ rates ] as starter for [ greedy1 ] simulation
100%|                                                                                | 2309/2309 [00:14<00:00, 162.00it/s]
Results = [0, 89, 802, 967, 327, 92, 24, 7, 1]
Score = 3.8423559982676485
Stop simulations? y/n
n
Choose mode: ['naive', 'greedy1', 'greedy2', 'greedy3', 'greedy4']
greedy2
Choose starter word or press [enter]

Using [ rates ] as starter for [ greedy2 ] simulation
100%|                                                                                | 2309/2309 [00:03<00:00, 630.86it/s]
Results = [0, 118, 976, 999, 173, 34, 6, 3, 0]
Score = 3.59246427024686
Stop simulations? y/n
n
Choose mode: ['naive', 'greedy1', 'greedy2', 'greedy3', 'greedy4']
greedy3
Choose starter word or press [enter]

Using [ rates ] as starter for [ greedy3 ] simulation
100%|                                                                                | 2309/2309 [00:37<00:00, 61.36it/s]
Results = [0, 178, 1012, 835, 214, 56, 13, 0, 1]
Score = 3.567778258986574
Stop simulations? y/n
n
Choose mode: ['naive', 'greedy1', 'greedy2', 'greedy3', 'greedy4']
greedy4
Choose starter word or press [enter]

Using [ rates ] as starter for [ greedy4 ] simulation
100%|                                                                                | 2309/2309 [00:08<00:00, 268.11it/s]
Results = [0, 337, 1202, 669, 88, 11, 2, 0, 0]
Score = 3.237765266349069
Stop simulations? y/n
y
```

```
(myenv) C:\Users\VATSALA NEMA\Documents\SEM 8\Info theory\CustomWordleAlgs-main\CustomWordleAlgs-main>C:/Apps/Conda/envs/my
env/python.exe "c:/Users/VATSALA NEMA/Documents/SEM 8/Info theory/CustomWordleAlgs-main/CustomWordleAlgs-main/InformationTh
eoryAlgorithm/NaiveGreedySimulation.py"
100%|| 2309/2309 [20:17<00:00,  1.90it/s]
[0, 8, 48, 82, 77, 20, 9, 2065]
Score =7.611953226504981

(myenv) C:\Users\VATSALA NEMA\Documents\SEM 8\Info theory\CustomWordleAlgs-main\CustomWordleAlgs-main>C:/Apps/Conda/envs/my
env/python.exe "c:/Users/VATSALA NEMA/Documents/SEM 8/Info theory/CustomWordleAlgs-main/CustomWordleAlgs-main/InformationTh
eoryAlgorithm/EfficientNaiveGreedySimulation.py"
100%|| 2309/2309 [06:42<00:00,  5.73it/s]
[0, 40, 535, 998, 472, 170, 59, 23, 12]
Score =4.227804244261585

(myenv) C:\Users\VATSALA NEMA\Documents\SEM 8\Info theory\CustomWordleAlgs-main\CustomWordleAlgs-main>C:/Apps/Conda/envs/my
env/python.exe "c:/Users/VATSALA NEMA/Documents/SEM 8/Info theory/CustomWordleAlgs-main/CustomWordleAlgs-main/InformationTh
eoryAlgorithm/Simulatorv1.py"
Choose mode: [naive] or [greedy1]
naive
Choose starter word or press [enter]

Using [ tares ] as starter for [ naive ] simulation
100%|| 2309/2309 [00:19<00:00, 119.16it/s]
Results = [0, 40, 535, 998, 472, 170, 59, 23, 12]
Score = 4.227804244261585
Stop simulations? y/n
n
Choose mode: [naive] or [greedy1]
greedy1
Choose starter word or press [enter]

Using [ rates ] as starter for [ greedy1 ] simulation
```

```
naive
Choose starter word or press [enter]

Using [ tares ] as starter for [ naive ] simulation
100%|| 2309/2309 [00:18<00:00, 123.58it/s]
Results = [0, 40, 535, 998, 472, 170, 59, 23, 12]
Score = 4.227804244261585
Stop simulations? y/n
n
Choose mode: [naive] or [greedy1] or [greedy2]
greedy1
Choose starter word or press [enter]

Using [ rates ] as starter for [ greedy1 ] simulation
100%|| 2309/2309 [00:14<00:00, 158.01it/s]
Results = [0, 89, 802, 967, 327, 92, 24, 7, 1]
Score = 3.8423559982676485
Stop simulations? y/n
```

Is this the optimal solution?

TLDR: NO

References:

1. https://arxiv.org/pdf/0903.1659.pdf
2. Solving Wordle using information theory
3. Wordle solving algorithms using Information Theory Universidad Complutense de Madrid