Kevin, Aaron, Kelley

Roadmap

1 Purpose & Overview

2 Individual Strategies

3 Combined Strategies

4 Insights & Conclusion

Wordle is a trending word puzzle game.

Goal: Guess a 5-letter word in 6 tries

We are looking to explore different strategies and determine which one is the best.

Examples

The letter **W** is in the word and in the correct spot.

The letter **I** is in the word but in the wrong spot.

The letter **U** is not in the word in any spot.

Insight into Wordle Data

10,658 possible guesses

2,316 possible answers

Sources: Web -Scraped from the NYTimes Website

Key Questions & Goals

1. What is the most effective strategy based on number of guesses, run time, and success rate?

2. What is the most efficient human centric strategy?

Goals

1. Create algorithms that mimic popular Wordle Strategies

2. Create metrics to rank strategies

3. Factor translation into real world application

Our Strategies

Tested Strategies:

- 'Entropy' of a word
- Overlapping Letter & Letter Frequency
- Exclude Common Letters

Why these strategies?

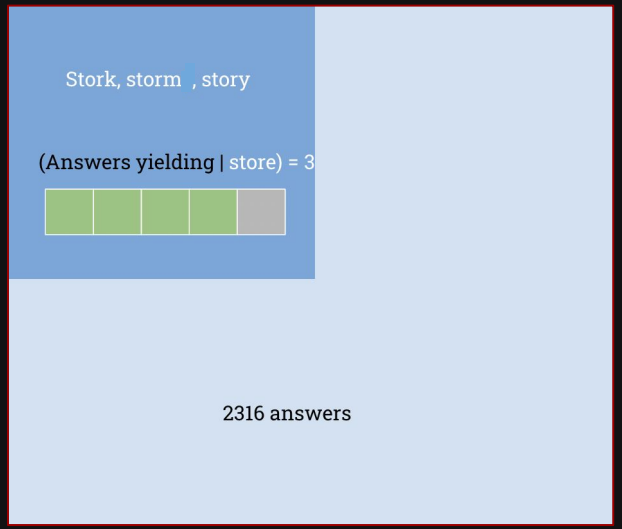
- Explore what machine learning can do
- Mixture of human & machine approaches
- Accessible to human players

Entropy (1 / 4)

- Entropy = “how much information an event gives”
- Unlikely events give more information
- Two greens is more unlikely than all greys
- Two greens tell us more

Entropy (2 / 4)

Entropy of S T O R E



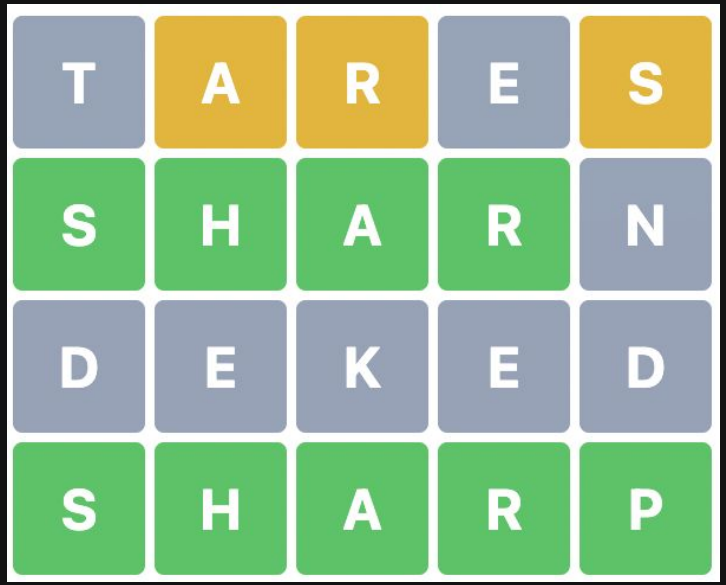
$$P(\frac{\text{Stork, storm , story}}{\text{All 2316 answers}}) = (\ ?\ )$$

Ent(S T O R E)

Prop. answer space: ½, ¼, etc.

Ent(S T O R E)=9.59

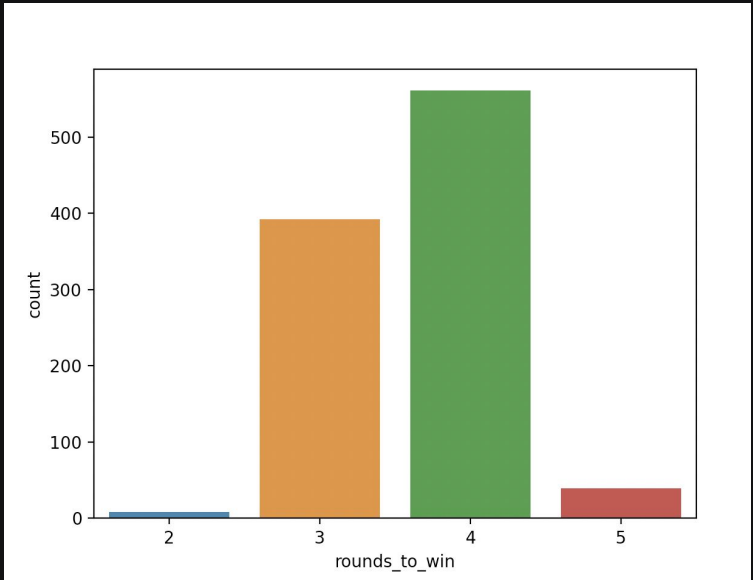
Entropy (3/4)



← Shard, Share, Shark, Sharp

1. Calculate entropies for guesses
2. Play word with max entropy
3. Narrow answer list
4. Re-calculate entropy

Entropy (4 / 4)



How well did it work?

Out of 1000 trials:

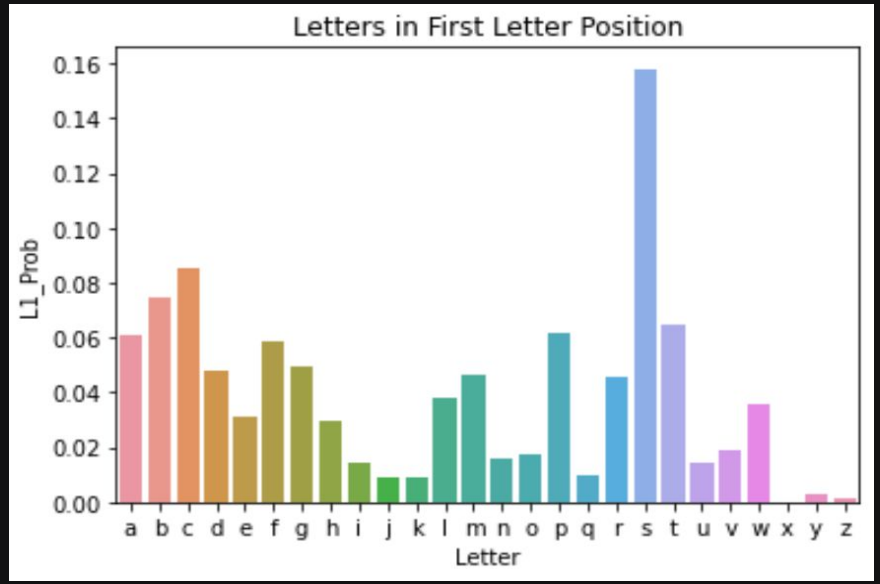
- Success rate of 100%
- Averaged 3.631 guesses
- Run-time = 3 hours

Letter Frequency (1/5)

What is the Strategy?

- A mixture of 'human' strategies and algorithms
- Algorithm filters out words without overlapping characters & ranked them based on letter frequency
- Next Guess = word with overlapping characters & highest probability

Letter Frequency (2/5)



How were frequencies calculated?

Ex) Probability of "CRANE"

- P("C"| 1st Position) = 0.0855
- P("R"| 2nd Position) = 0.115335
- P("A"| 3rd Position) = 0.132613
- P("N"| 4th Position) = 0.78618
- P("E"| 5th Position) = 0.183153

Assuming independence of position,

P("CRANE") = 1.8836364893999484e-05, or 0.0000188

Letter Frequency (3/5)

- **Ex) If Target = "Story"**

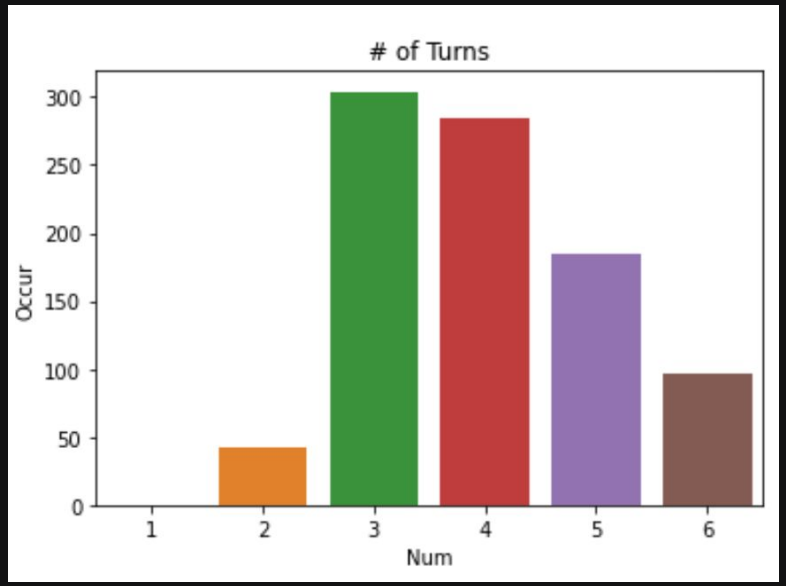
Guess 1: 

  - **Step 1: Generate List of Possible Words**
    - Algorithm gathers words with:
      - No "D" or "W"
      - "O" but not in 2nd position
      - "R" and "Y" in 4th and 5th position
    - Result - List 1: ['glory', 'ivory', 'ovary', 'story']

Letter Frequency (4/5)

- **Step 2: Generate Probability**
  - Algorithm calculates the probability of words & ranks in order of size
    - "Story" - 5.7220497802773l8e-06
    - "Glory" - 4.6932503254243 92e-06
    - "Ovary" - 1.5710987608141618e-07
    - "Ivory" - 1.0354997473356867e-07
  - Algorithm chooses word with highest prob
    - Guess2 = "Story"

Letter Frequency (5/5)



How well did it work?

Out of 1000 trials:

- Success rate of 91%
- Average Number of 3.79 turns
- Run-Time: 70 seconds

Excluding Common Letters (1 / 4)

What is the Strategy?

- Another 'human strategy'
- Algorithm makes three initial guesses with each guess sharing no letters
- Fourth Guess and beyond, the algorithm takes the information from the previous guesses to filter out the remaining possible answers

Excluding Common Letters (2 / 4)

Ex) If Target = "Story"

Generate 3 guesses with no shared letter between each other

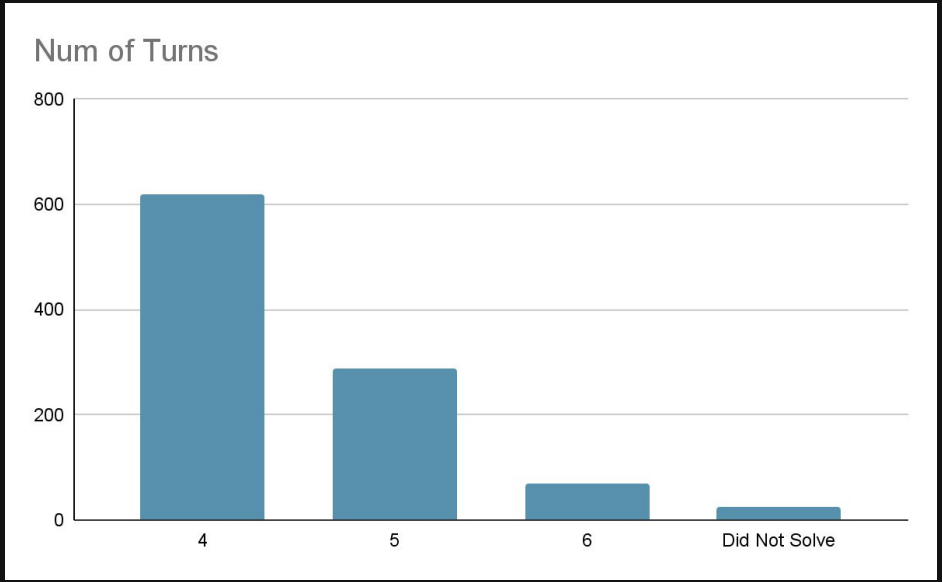
Taking into account the results of the first three guesses the algorithm determines that “Story” is the only possible answer.

Excluding Common Letters (3 / 4)

In the previous example after three guesses the algorithm was able to narrow the possible answers down to one answer.

There are many cases where this is not possible and at the fourth guess the algorithm will have a list of a few options. In this case the algorithm will pick one at random and then use those results to filter out more answers until it finds the right answer.

Excluding Common Letters (4 / 4)



How well did it work?

Out of 1000 trials:

- Success rate of 97.6%
- Average Number of 4.33 turns
- Run-Time: 7 minutes

Combining Strategies (1 / 4)

Q-Learning

- We know the rules, we don't know how to best play

- Machine learns both rules and optimal play

- Environment
- Actions
- States
- Rewards

Q(state, action) = reward + discount_r(future reward)

Qnew = Qold + learning_r(Q(state, action) - Qold)

Combining Strategies (2 / 4)

Rewards:

- Win=30, Lose=15

States (S_t)

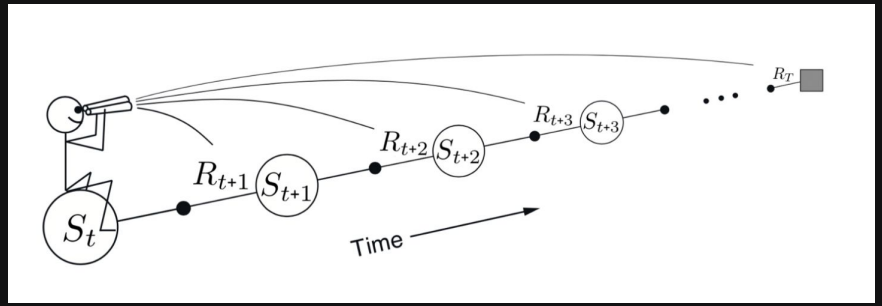
- Total unique (#Green_pos, #Yellow_pos) observed at time t in the game
  - E.g. (2,3): [T1, U2, U4, C3, H5]

Epsilon-Greedy:

- Epsilon: 0.1 w/ decay; initial Q-values = 10

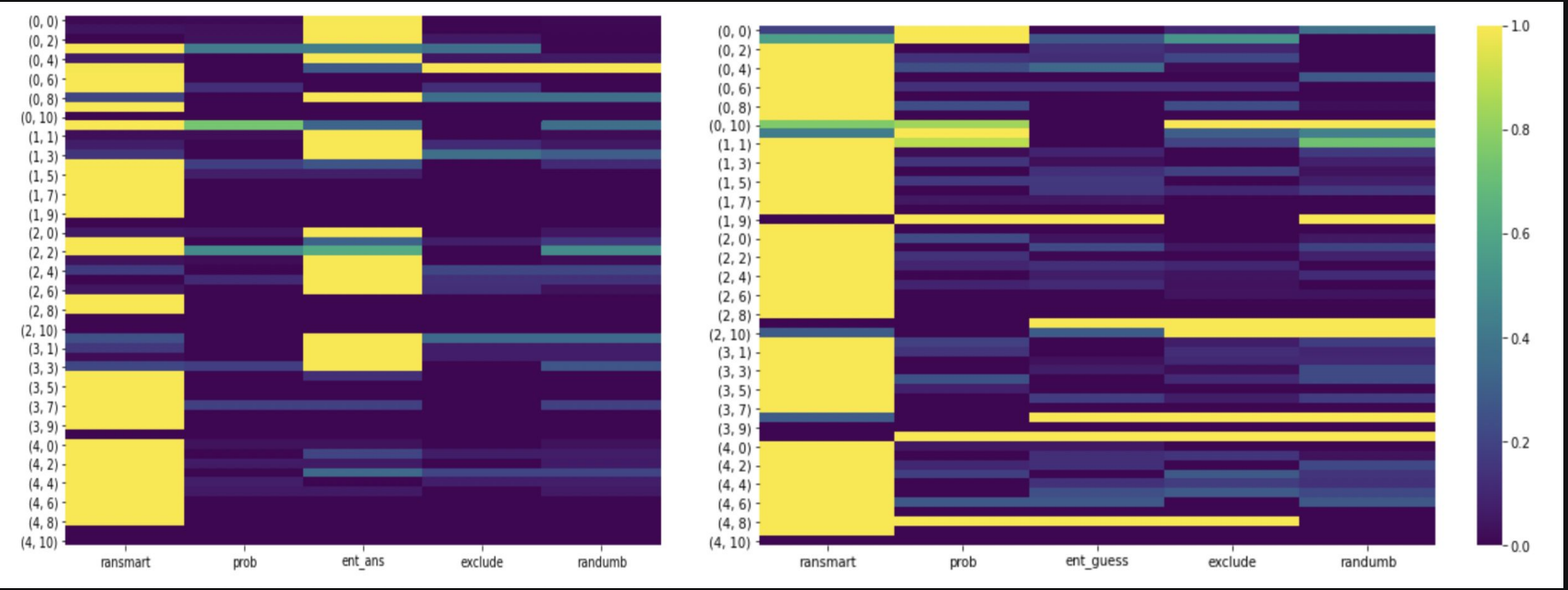
Algorithm:

- Updates Q(state, action) 3 guesses after
- Discounted reward through 3 guesses



Combining Strategies (3/4)

Version 1



Version 2

Combining Strategies (4/4)

	Roadmap (v1)	Roadmap (v2)
0 1	Entropy	Letter Frequency
1 2	Entropy	Letter Frequency / Make any guess based on possible answers
2 2	Make any reasonable guess based on possible answers	

Future Improvements

- **Address the "independence between position and letter" assumption**

- **Switch strategies after 'n' guesses rather than every turn**
  - Ex. the 'Exclude Common Letter' strategy works a lot better if it's run twice rather than once

Conclusion & Insights

Early stages of the game:

- Entropy works best
- Letter frequency may have slight advantage

Late game:

- No clear difference
- Simply tracking what word is / isn’t possible is often the cleanest solution