WORDLE

Optimising Solution Algorithms for n-Blank Guessing Puzzles

Group 31: Shivani Verma, Suryansh Singh, Tejal Barnwal, Sahil Malik

TABLE OF CONTENTS

01

INTRODUCTION

Problem statement and description of Wordle

02

FORMULATION

Wordle problem Formulation in characteristic and information matrices

03

APPROACH

How to score guesses. Set covering and decision trees.

04

ALGORITHM

Scoring using Information Entropy. Wordle Demo.

05

OPTIMALITY

Comparing Algorithms: Max Min score vs Expected score approach

06

FUTURE WORK

Shortcomings and Possible improvements.

01

Introduction

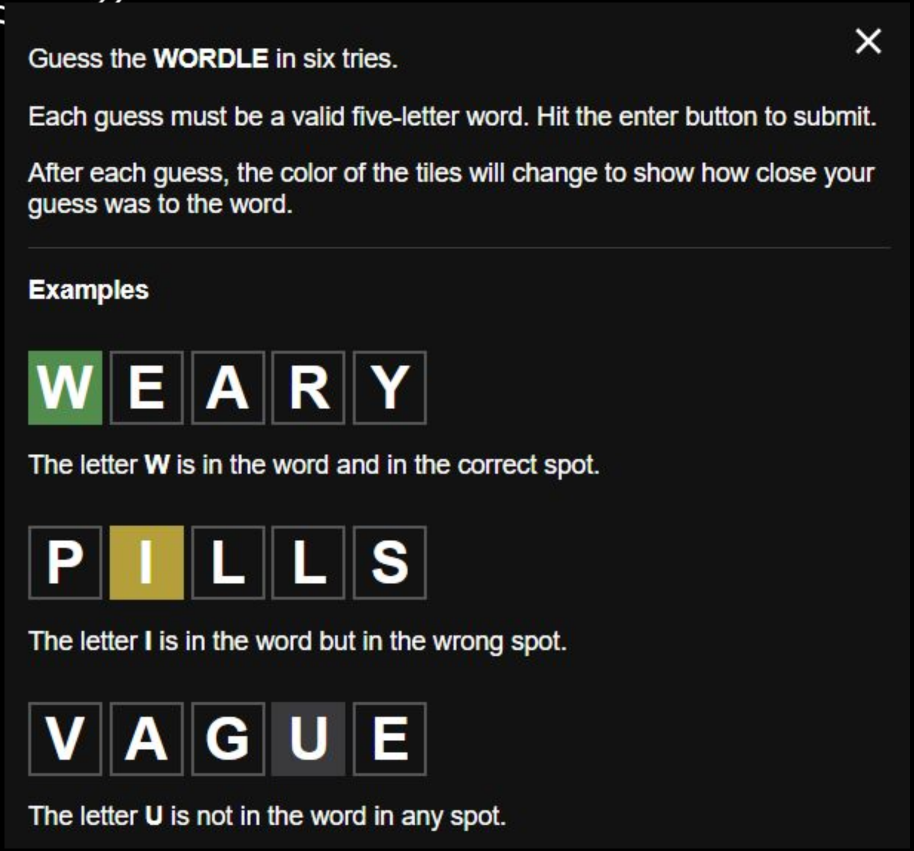
Introduction: n-blank puzzle

We have an n blank guessing puzzle with m tries, that can take sequences like $\mathbf{x} = (x_1, x_2, ..., x_n)$ such that $\forall\ x_i \in F$, where F is the finite set of elements that make up all sequences. Let $F^n = U$ be the set containing all possible sequences $\mathbf{x}$ and $\mathrm{G} \subseteq U$ is the feasible set of possible solutions to the guessing game.

We try this on the popular game Wordle.

Here F is the english alphabet,
U is the set of every sequence of letters (26^5)
G is the set of all allowed words you can guess.

The Puzzle: WORDLE


Guess the **WORDLE** in six tries.

Each guess must be a valid five-letter word. Hit the enter button to submit.

After each guess, the color of the tiles will change to show how close your guess was to the word.

Examples

WEARY

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

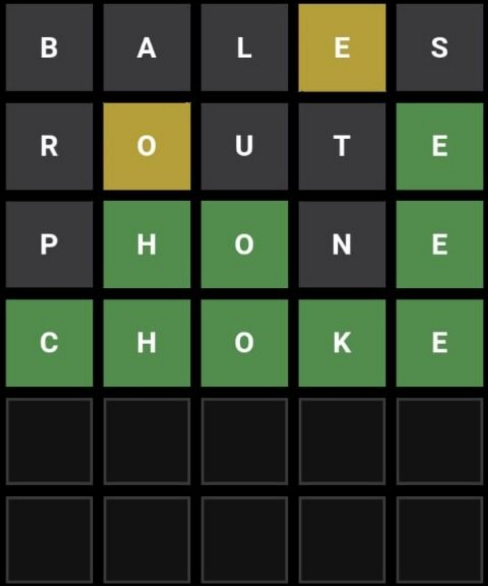
PILLS

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

VAGUE

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

We see that every guess has a possible set of outcomes which gives us clues as to what the solution could be.



Wordle Play:
An Example

Here the solution is CHOKE

We see that the guess set is the superset of the solution set and can be used to explore the solution set.

Guess Set

(12989 words)

G = { TABLE, SOARE, FLAKE, SHARE, SWIMS, RANCE, . . . , SALET, POINT, QUICK, . . . , WEARY, WAQFS, ZEBRA }

All guess sequences $\mathbf{g}_i$ *belong to G*

$$\bar{g}_i = (g_1, g_2, \ldots, g_n) \in G$$

$$S \subseteq G$$

Solution Set

(2315 words) (feasible set)

S = { TABLE, FLAKE, SHARE, SWIMS, POINT, QUICK, . . . , WEARY, ZEBRA }

All solutions sequences $\mathbf{s}_i$ *belong to S*

$$\bar{s}_i = (s_1, s_2, \ldots, s_n) \in S$$

02

Problem Formulation

1. Characteristic Matrix (A)

Every word is assigned a characteristic matrix.

A characteristic matrix is a (26x6) matrix that defines the word:

1. the letters that occur
2. their position.

Elements not shown are assigned the value -1

TABLE

A	1	-1	1	-1	-1	-1
B	1	-1	-1	1	-1	-1
C	-1	-1	-1	-1	-1	-1
D	-1	-1	-1	-1	-1	-1
E	1	-1	-1	-1	-1	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮
L	1	-1	-1	-1	1	-1
⋮	⋮	⋮	⋮	⋮	⋮	⋮
T	1	1	-1	-1	-1	-1
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Z	-1	-1	-1	-1	-1	-1

QUICK

A	1	-1	-1	-1	-1	-1
B	1	-1	-1	-1	-1	-1
C	1	-1	-1	-1	1	-1
⋮	⋮	⋮	⋮	⋮	⋮	⋮
I	1	-1	-1	1	-1	-1
J	-1	-1	-1	-1	-1	-1
K	1	-1	-1	-1	-1	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Q	1	1	-1	-1	-1	-1
⋮	⋮	⋮	⋮	⋮	⋮	⋮
U	1	-1	1	-1	-1	-1
⋮	⋮	⋮	⋮	⋮	⋮	⋮

2. Information Matrix (P)

Let’s say we guessed SALET, and wordle gave us the following:

The information Matrix (26x6) would be:

A	1	0	1	0	0	0
B	-1	-1	-1	-1	-1	-1
⋮	⋮	⋮	⋮	⋮	⋮	⋮
E	1	0	0	0	-1	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
L	1	0	0	-1	0	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
S	-1	-1	-1	-1	-1	-1
T	1	0	0	0	0	-1
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Z	-1	-1	-1	-1	-1	-1

3. Decision Matrix (M)

It is given by multiplying the elements of the characteristic and information matrices, (also 26x6)

M = $[m_{i,j}] = [a_{ij} \times p_{ij}]$

4. Constraints

1. All words are FIVE lettered
2. Given set S containing all the feasible solutions and set G containing all the feasible guesses
3. Constraints according to green, yellow and grey:

   Let solution be $(s_1, s_2, s_3, s_4, s_5) \in S$ and let the guess be $(g_1, g_2, g_3, g_4, g_5) \in G$

   a. <u>Green</u>: $g_i = s_j$ and $i = j$ such that $\forall\, i, j \in \{1, 2, 3, .., n\}$

   b. <u>Yellow</u>: $g_i = s_j$ and $i \neq j$ such that $\forall\, i, j \in \{1, 2, 3, .., n\}$

   c. <u>Grey</u>: $g_i \neq s_j$ such that $\forall\, i, j \in \{1, 2, 3, .., n\}$

5. Decision Variables

1. x_w: Given characteristic matrix $M = [m_{ij}]$,
   if $m_{ij} < 0$ for any i,j, then $x_w \leq 0 \ \forall \ w \in G$
   (x_w checks feasibility of guess w)
2. s_w: Score of each word w calculated based on Scoring Algorithm $\forall \ w \in G$
3. y_w: If s_w is maximum, then $y_w = 1$; otherwise y = 0
   (If y = 1, we are making a guess)
4. If $m_{ij} \geq 1 \ \forall \ i \in \{1, 2, \ldots, 26\}, j \in \{1, 2, \ldots, 6\}$, then break
   (Stopping condition; solutions found)

6. Objective Function

1. Local Objective Function: Finding the argument word that has the maximum score for the next guess

$$w \ = \ argmax(s_w) \ \forall \ w \in G$$

2. Global Objective Function: Minimize the total number of guesses needed

$$min \sum_{w} y_w \ \ \forall \ w \in G$$

03

Approach

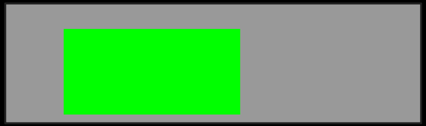
SET CUTTING

IDEA: To minimise #tries we want to minimise our path to the solution, wherever in the set it might be.

This leaves 49 words remaining

This leaves only 1 word: ARENA

196 words left for this possibility



Feasible section of the solution set

Discarded section of the solution set

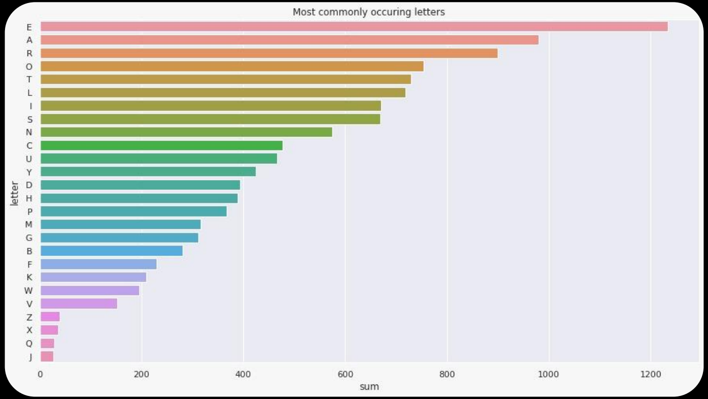
So if any guess/clue shrinks our solution set to a smaller pool, probability of guessing the solution in the next try increases

HOW TO SCORE OUR GUESSES

1. Letter Frequency

Captures the occurrence and position information of each individual alphabets in the solution set

(SET COVERING algorithm)



	A	B	C	D	E	F	G	H	I	J
A	70	107	179	130	362	66	110	123	142	7
B	107	13	29	38	119	7	25	26	62	2
C	179	29	29	37	171	25	13	118	115	3
D	130	38	37	22	179	18	42	29	123	2
E	362	119	171	179	172	83	131	136	232	9

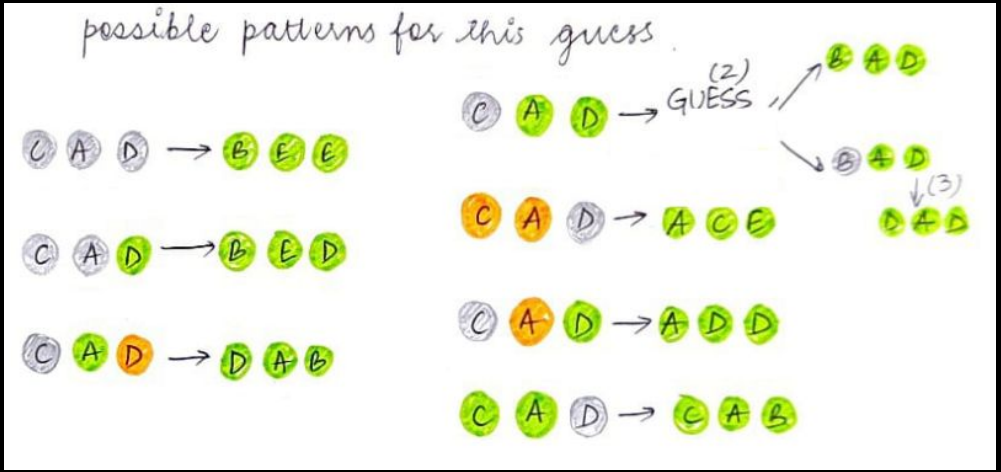
2. Covariance

Encompasses occurrence of any two alphabets and returns number of words that contain the corresponding pair

3. Using Decision trees

Entropy:
Is the expected information when you play a certain guess.
If each outcome E_i of playing a guess *g* has probability p_i of occurring then

Entropy, $$E = \sum_{i}^{n} p \times log_2(1/p)$$



For the guess CAD, assuming only 3 letter words formed by {A,B,C,D,E}

04

Algorithm

Using Information Theory: Entropy

Scores for 1st iteration

$$E = \sum_{i}^{n} p \times \log_2(1/p)$$

guess	score = entropy (bits)
SOARE	5.8860
ROATE	5.8828
RAISE	5.8779
RAILE	5.8657
REAST	5.8655
SLATE	5.8558
CRATE	5.8349
SALET	5.8346
IRATE	5.8314
TRACE	5.8305

$$\log_2(1/p)$$

guess	score = min info (bits)
RAISE	3.784479055
RAILE	3.742168251
SOARE	3.66109664
IRATE	3.576883636
ROATE	3.569466164
SLATE	3.388893919
SALET	3.388893919
REAST	3.350247991
CRATE	3.234281973
TRACE	3.234281973

CODE DEMO

Colab Link

05

OPTIMALITY

Proof of correctness

There are 2315 words in the solution set.
Information needed for get to any single word, we need 2315 divisions of the solution set

∴ I = -log2(1/2315) = 11.17 bits

With any guess with average entropy of 5.86 bits (best 10 guesses) we would need 2-3 iterations to reach 11.17 bits in most scenarios.

Even with minimum possible information from these guesses, an average of 3.49 bits, 3-4 iterations will complete 11.17 bits for most, and in 5 iterations for any word in our solution set. This was found computationally.

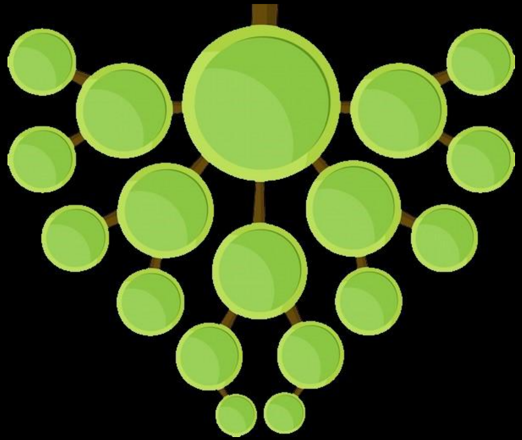
Our demo program (with minor cutbacks) has a worst performance of 6 tries.

Justification for this Scoring Algorithm

It is based on shortest path approach for an unknown node.

It accommodates letter frequency and letters occurring togethers

It also accounts for probability of an outcome before trying the guess, as opposed to just post clue analysis.



IS IT OPTIMAL?

This algorithm took on average 3-4 tries to guess the solution, which was better than all the other ideas we implemented.

So it’s just the best one we found.

06

FURTHER IMPROVEMENTS

Further Improvements

- **Scoring**
  An improved scoring could include a combination of max-min and expected information heuristic
- **Repeated Letters**
  We did not Include guesses with repeated letters due to our limited programming knowledge. Including this would give a better performance.
- **Enumeration and Time Complexity**
  The computations for the scoring are the major bottleneck as of now. However, they can be solved using better data structures.
- **Further Use for the algorithm**
  Developing a generalised, robust, and generalised code for use outside Wordle

References

1. Wordle - The New York Times
2. Solving Wordle using information theory
3. Wordle Solver
4. Mathematical optimization over Wordle decision trees -- Laurent's notes
5. Wordle-solving state of the art: all optimality results so far -- Laurent's notes
6. Absurdle @ Things Of Interest
7. Quordle

THANK
YOU