Problem Chosen	2023 MCM/ICM Summary Sheet	Team Control Number
C		2322298

Our solutions come in 2 main parts — predictions on the number of results and the establishment of the model of distribution & difficulty.

For the number of reported results prediction, we tried to predict result-counts of whole weeks to reduce impacts caused by the differences between weekdays and weekends. We assume that the first 20 weeks (January 7, 2022 to May 28, 2022) are the period of time that the game is trending, which results in great rises and declines in result-counts. After 20 weeks, the curve of result-counts stabilizes and can be characterized by a fractional function. And for March 1, 2023, which is a Wednesday, we use the normal distribution to obtain an estimation of [20935, 23909], with a confidence of 87.2%. We didn't spot the relationship between hard-mode-result-counts and the word of that day, and it is reasonable — people need to decide whether to turn on hard mode before their very first guess. As people know nothing about the word before the game, turning on/off hard mode depends more on yesterday's result.

Another main part is to build the model of distribution & difficulty. To make things easier, we can regard the distribution of results as a 7-dimensional vector, and call the result distribution of word s from the provided dataset "human-distribution", denote as $\mathbf{h}(s)$. Here's the procedure of how our model works:

- Suppose that we want to get the distribution of results and the difficulty of a specific word s_0.
- We first put it into our man-like wordle solver, which obtains distribution of results by simulating how a human "wordles" the word s_0 thousands of times and returns the so-called "machine-distribution" (denote as $\mathbf{v}(s_0)$).
- Along with other 359 words from the provided dataset, we run k-means clustering on these 360 machine-distributions, where given words' machine-distribution ($\mathbf{v}(s_1)$, $\mathbf{v}(s_2)$, $\cdots$, $\mathbf{v}(s_{359})$) are obtained beforehand.
- We will obtain 5 clusters, where each cluster gathers together words with similar attributes. Find the cluster that s_0 belongs to, and denote the set of all words in that cluster as S.
- Use a heuristic algorithm to find a matrix T, such that $\frac{1}{|S|}\sum_{s\in S}\|\mathbf{v}(s)T-\mathbf{h}(s)\|_2^2$ is minimized.
- Then, $\mathbf{v}_0=\mathbf{v}(s_0)T$ is the final prediction of distribution of the word s_0.
- And if we write $\mathbf{v}_0=(v_1,v_2,v_3,v_4,v_5,v_6,v_7)$, we define $d_0=10v_7+\sum_{k=1}^{6}kv_k$ as the difficulty of the word S_0.

Our estimation of distribution and difficulty for "eerie" are (0, 2, 11, 27, 34, 21, 5) and 4.91.

We notice that these attributes of the word might affect the word's distribution & difficulty: familiarity to people, existence of duplicate letters, existence of rare consonants, existence of rare letter combination, and existence of similar structured.

Contents

1 Building the Model of Distribution & Difficulty

In this section, we will introduce how we build the model which takes a word and gives the distribution of the results. We first write a program that provides us with "machine-distributions". Though it is called "machine-distributions", we have done our best to make it perform like a human. Then we distribute words into clusters and deal with each cluster separately. We use linear algebra to "transform" machine-distributions into human-distributions, where we find a magic matrix T and multiply all machine-distributions $\mathbf{v}$ by it. With the back of linear algebra, we can obtain an "accurate" distribution estimation.

As for difficulty, we link distribution directly with it, which in our mind can reflect the level of difficulty. The word itself has something to do with difficulty, for sure, but its attribute also contributes to the distribution of results. In other words, our model of difficulty is based on the model of distribution. And here's the relationship between these three concepts.

$$\text{Word} \underset{\text{partly reveals}}{\overset{\text{defines}}{\rightleftharpoons}} \text{Distribution} \underset{\text{partly reveals}}{\overset{\text{reflects}}{\rightleftharpoons}} \text{Difficulty}$$

1.1 The Man-like Wordle Solver

In this subsection, we will follow the procedure of how our man-like wordle solver is built, from how it is established to somehow resemble humans.

We didn't write about every adjustment we made to the solver to make it play like a human, because many of these adjustments are very specific and are strongly related to attributes of a word. However, to make the whole paper shorter, we discussed several word attributes that we think might contribute to distribution and difficulty in 1.3.

1.1.1 How to Solve a Wordle Properly

We want to write a program that simulates the process of mankind solving a Wordle problem.

Thanks to 3Blue1Brown, who made a video about how to solve wordles efficiently months ago, we are able to build a machine that is capable of solving wordles "faster" than humans.

We need information theory to help us pick the proper word to solve wordles smartly. After we type a word in the wordle game, we would get feedback, and the feedback depends on both the word we type (denote it as S) and the answer word (denote it as A), say $f_A(S)$ is the feedback (which is 5 green, yellow, or gray blocks). But, of course, we don't know the answer beforehand, so we need to consider every possible answer $A_{1,2,\cdots}$, and need them to help us evaluate the benefit we will get after typing S.

Let's denote the benefit we'll get from S as $b(S)$, where the higher the $b(S)$ is the more you're willing to type S for the wordle guess. And then assume that you get feedback $f = f_A(S)$ after typing

S, and you'll know that potential answers are inside set $\mathcal{A}(f)$, where: ($\mathcal{V}$ denotes the set of 5-letter words that are still possible to be the answer before typing S)

$$\mathcal{A}(f) = \{A \mid f_A(S) = f, A \in \mathcal{V}\}$$

Which is the set of all words that will give you the same feedback f. We can define an equivalence relation such that $A \sim B \iff f_A(S) = f_B(S)$, and its reflexivity, symmetry, and transitivity can be easily proved. Thus, one can tell that $\mathcal{A}(f)$s are the equivalence classes.

Think about after typing S and obtaining the desired feedback f, which kind of $\mathcal{A}(f)$ would we expect? could its size be too big? No, then the remaining number of potential answers would be too large, leaving us with a hard game. Then are we expecting small $|\mathcal{A}(f)|$? No, either. Although $\mathcal{A}(f)$ with a small size can help us exclude lots of potential answers, the chance that the feedback f leads you to $\mathcal{A}(f)$ is small, which is $|\mathcal{A}(f)|/|\mathcal{V}|$.

Think about normal divide-and-conquer algorithms, where we evenly divide the "task" so that we can obtain the best worst-case and average-case performance. We do a similar thing here, we what the sizes of different $\mathcal{A}(f)$ to be uniformly distributed, and we can write our requirement in a mathematical way: find a proper S, such that the following expression is minimized: (F denotes the set of all possible feedbacks, where $|F| \leq 3^5$)

$$\max_{f \in F}\{\mathcal{A}(f)\} - \min_{f \in F}\{\mathcal{A}(f)\}$$

Inspired by 3Blue1Brown's video, we can also describe this requirement in an information theory way. Define the possibility that we will encounter feedback f:

$$p(f) = \frac{|\mathcal{A}(f)|}{|\mathcal{V}|}$$

And then define the entropy decrease after obtaining feedback f, which is the amount of information that f tells us (higher the better):

$$I(f) = -\log_2(p(f)) = \log_2\left(\frac{|\mathcal{V}|}{|\mathcal{A}(f)|}\right)$$

Then, of course, we want to find such a word S, that its expected entropy (amount of information) given to us through feedback is maximized, which we can derive the formula for $b(S)$:

$$\begin{aligned} b(S) = \mathbb{E}[f] &= \sum_{f \in F} p(f) I(f) \\ &= \sum_{f \in F} \frac{|\mathcal{A}(f)|}{|\mathcal{V}|} \log_2\left(\frac{|\mathcal{V}|}{|\mathcal{A}(f)|}\right) \end{aligned}$$

With this criterion in mind, choosing the word with the highest $b(S)$, gives us a basic wordle solver that could find out the answer within about 3 or 4 guesses.

1.1.2 Considering Familiarity

We humans never play wordle like the machine does, because we don't calculate possibilities and entropies and we sometimes stick to our familiar word. And we don't type words into wordle, which seldom occur to us but contain a huge amount of "information", so we need to lower the priority of words that are unfamiliar to humans.

We obtain the frequency of words from WordFrequencyData, provided by the Wolfram Language. Most of the 5-letter words, as a potential wordle answer, have a frequency ranging from 10^{-7} to 5×10^{-5}. And we don't think that there exists a linear relationship between frequency and familiarity, which reveals that wordle words aren't "naïve" words.

Our idea is that we find a logarithm function, which maps a frequency of 10^{-7} to a familiarity of 20, and 10^{-5} to a familiarity of 80, because we think that most people can't recognize words with a frequency less than 10^{-7}, and words with a frequency more than 10^{-5} can be easily come up with. Suppose the mapping between frequency and familiarity is (constant b and c are to be determined):

$$p(x) = \min\left\{100, \log_b(cx)\right\}$$

Then, $p(10^{-5}) = \log_b(c \times 10^{-5}) = 80$ and $p(10^{-7}) = \log_b(c \times 10^{-7}) = 20$, and these give us enough information to solve for b and c:

$$c = b^{80} \times 10^5 = b^{20} \times 10^7 \implies b^{60} = 10^2$$

We can easily obtain that $b = \sqrt[60]{100}$, $c = 10^{23/3}$, by approximation, we set $b = 1.0797751623277096$, $c = 46415888.33612776$. The fun fact is that among the 2309 possible wordle answers, the average familiarity is around 65.825.

To involve familiarity, we denote $g(S)$ as the familiarity of word S, and we need to update the benefit function a little bit:

$$b(S) = g(S) \times \sum_{f \in F} p(f) I(f)$$

By far, our machine's wordle performance dropped and it may guess the answer as a common word for the second and third try and may need more tries for words that are unfamiliar to humans.

1.1.3 Involving Randomness

Efforts made above can't directly give us the distribution of results by "wordle-ing" the same answer word multiple times, but it could be achieved by involving randomness during the process of picking the to-be-typed word.

Instead of choosing the word S with the highest $b(S)$, we want our machine randomly pick a word from the top list ordered by $b(S)$. Doing so makes the machine's behavior more like a human's because we can't always come up with the best one, most of the time, we just type the rather good one that jumps into our mind.

Randomness enables our machine to produce a rough distribution that has a similar shape compared to real distributions. But its output is still far from a so-called "accurate" estimation.

1.1.4 Thinking in Different Ways

It comes to our mind that different people play wordle in different manners.

Some people make their choice on familiarity only. They can be kids, people who haven't mastered the game yet, or just don't want to spend too much time in this game. And their benefit function for a word S totally depends on familiarity:

$$b_1(S) = g(S)$$

Some other people make their choice "wisely", which means, they greatly emphasize how much information they can get from a certain word. They may be pro players or just people who solve daily wordle with their well-designed bots. Their benefit function for a word S mainly depends on the "information" that the word contains:

$$b_2(S) = \sum_{f \in F} p(f) I(f)$$

While others, like most of us, consider both familiarity and entropy decrease that a word can bring. And the benefit function is the revised one:

$$b_3(S) = g(S) \times \sum_{f \in F} p(f) I(f)$$

We suggest our machine think in different modes during each step, and we modified how it picks the word to type next: first, find out the top n_1 words according to $b_1(S)$; second, find out the top n_2 words according to $b_2(S)$; third, find out the top n_3 words according to $b_3(S)$; forth, randomly pick out the word among these $n_1 + n_2 + n_3$ words.

We did lots of experiments on the distribution of n_1, n_2, and n_3. Finally, we set $n_1 = 0$, $n_2 = 25$,

$n_3 = 25$, for its outputs are closer to the real distribution collected by the wordle organizer. Surprisingly and interestingly, it seems that just choosing the most "familiar" word seems to drive it even further from how an actual human would behave.

1.2 Quantifying Difficulty

Another important task apart from modeling the distribution of results is to quantify the difficulty of a word. The first method that occurs to us is to calculate the expected tries with respect to the distribution, and this method is proved to be efficient and powerful.

To make things simpler, we treat the distribution of a word as a 7-dimensional vector:

$$\mathbf{v}(S) = (v_1, v_2, v_3, v_4, v_5, v_6, v_7)$$

or simply just $\mathbf{u}$ or $\mathbf{v}$ rather than $\mathbf{u}(S)$ or $\mathbf{v}(S)$ when causing no ambiguity. And the difficulty of a word S is simply:

$$d(S) = 10v_7 + \mathbb{E}[v_k, k \leq 6] = 10v_7 + \sum_{k=1}^{6} kv_k$$

We place a coefficient of 10 for v_7 to emphasize the failure rate. Without doubts, words are really difficult when people can't figure them out within 6 steps.

1.2.1 K-Means Clustering

As we have regarded distributions as vectors, then we can use k-means clustering algorithm to assign these vectors into k different groups or "clusters". These clusters not only enable us to explain why $d(S)$ is a valid difficulty modeling but also offer us a better insight into attributes of words when looking closely at words in the same clusters.

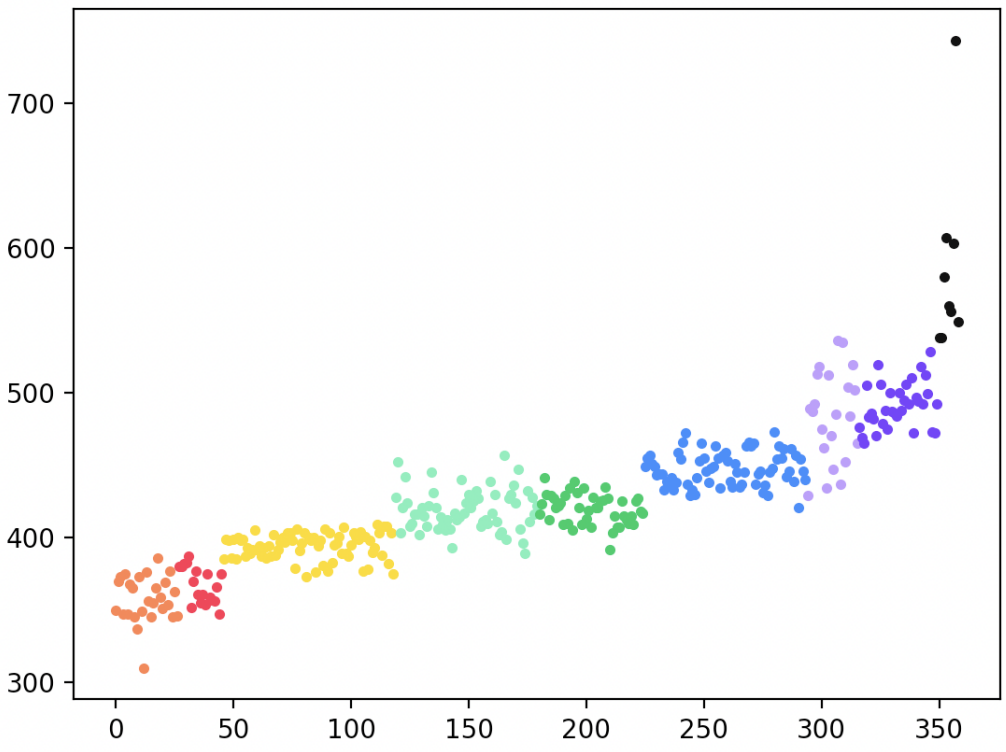

Figure 1: 10 clusters, sorted by $d(S)$

When we divide 359 human-distributions into 10 clusters (as shown in Figure 1), we can clearly find 6 difficulty levels. The **reds** are the easiest, then the **yellow**, follow by **green** and **blue**, after that **purples** are hard, and, finally, the **blacks** is extremely hard. Here, different clusters at the same difficulty level are plotted in a similar color but with slightly different shades, just as the difference between **light purple** and **normal purple**.

Recall that vectors in the same cluster have something to share in common, because their positions are alike or close to each other, according to the algorithm. And Figure 1 tells us that these similar vectors actually have similar difficulties within that cluster. This phenomenon shows that $d(S)$ is a good enough difficulty modeling. Such good lawyering is clearer with 5 clusters:

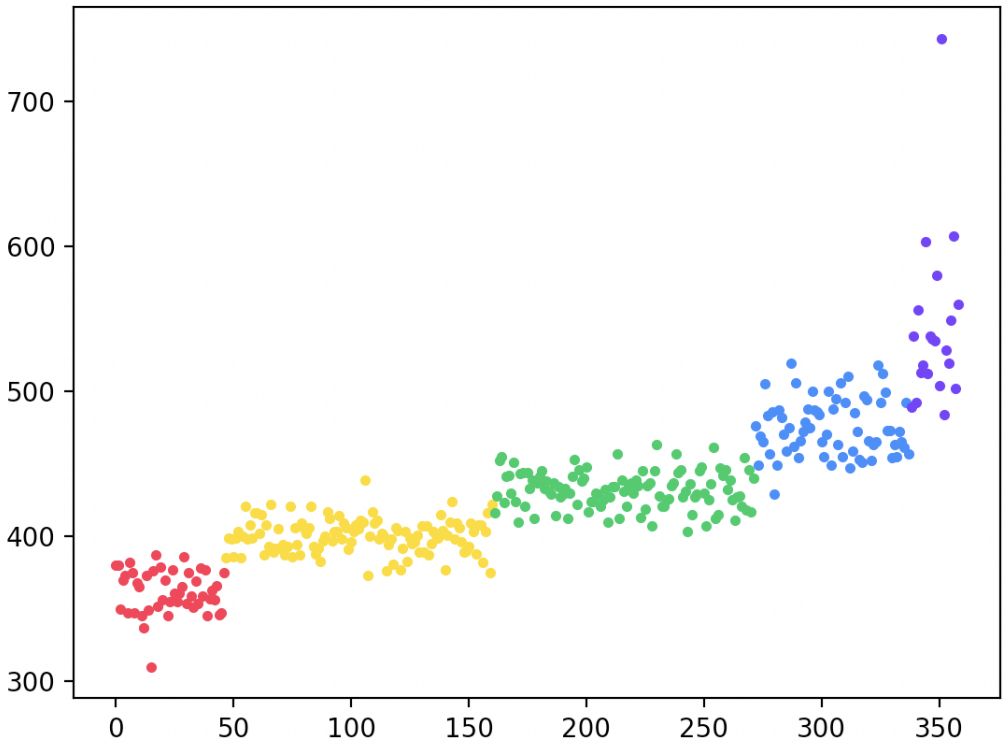

Figure 2: 5 clusters, sorted by $d(S)$

1.3 Finding Attributes of Words

With the help of clustering and by comparison between machine-distributions and human-distributions, we are able to notice several attributes of words that directly contribute to the distribution of results of that word, and indirectly to the difficulty.

K-means clustering does actually organize words with similar attributes together. You can find words with similar structures but different difficulties in the same cluster. For example, both “night” (point (107, 439) in Figure 2) and “light” (point (56, 421) in Figure 2) are in the yellow cluster, and “night” has the highest difficulty in that cluster which makes it “pop-out” while “light” is relatively easier. A similar situation goes for “catch” and “watch”, and “train” and “trash”. And more attributes would be discussed in the following.

We also used these findings to improve our man-like wordle solver, by making it ”sillier” when recognizing attributes described below. In the linear algebra optimization part, we use different linear transformations for different attributes (clusters of words) to make machine-distributions resemble human-distributions even more, mathematically.

1.3.1 The Importance of Being Common

When a word is pretty common to people, it has a great opportunity to have a lower difficulty. For example, “train”, “point”, “dream”, “rainy”, and “third” are some representatives.

On the other hand, when having a quite low familiarity, it might be some of the extremely hard-to-guess words. Several good examples are, “parer” (least common and hardest, has a familiarity of 5 and difficulty 743), “trite”, “foyer”, and “cinch”.

1.3.2 Duplicate Letters

Whether or not a specific letter has occurred several times in a word matters a lot. People tend to believe that they have omitted some consonants when getting only a few green and yellow after a few rounds. Furthermore, only a few people are willing to “sacrifice” one or two tries to test for duplicate letters, while others just keep trying to test for other “white” (never typed) consonants, believing that testing for duplicates is a waste for trial, or, at least, not informative.

With this mindset, humans’ performance in words containing duplicate letters worsens. We believe that most uncommon words with duplicate letters lay in the **blue** cluster among all five clusters (Figure 2), because 52% of the words in the **blue** cluster are words with duplicate letters, and the remaining of them are words containing rare consonants.

1.3.3 Rare Consonants

Wikipedia shows us that these are the top 8 rare consonants:

Letter	Frequency
Q	0.19%
J	0.21%
X	0.27%
Z	0.44%
W	0.91%
K	0.97%
V	1.00%
F	1.40%

Table 1: Relative letter frequency in English dictionaries (low)

It literally means that we need to try all letters before we can hit one of them unless the answer is pretty familiar to us. it takes us at least 5 tries to obtain the attendance of each letter, so words with rare letters are also difficult to solve.

On the other side, guessing the most common letters in the first shot can be helpful. And interestingly, the 5 most common English letters formed a reversed “raise”, which is also the first guess our man-like wordle solver used to use because it thinks it is the most informative.

Letter	Frequency
R	7.3%
A	7.8%
I	8.6%
S	8.7%
E	11%

Table 2: Relative letter frequency in English dictionaries (high)

1.3.4 Similar Structured Words

Another interesting point we’ve found when observing the machine-distributions is that we found that some common words such as “shake” are somehow not easy (difficulty: 489, which is really high). But we lately realized that “shake” shares the same “sha?e” structure with: “share”, “shave”, “shade”, “shame”, and “shape”. Their only difference is the consonant replacing the “?” in “sha?e”. The only thing you can do after figuring out this word structure while playing wordle is to guess which consonant it is and you have 6 candidates, leading you to a higher failure rate.

1.3.5 Rare Letter Combination

The uncommon letter combination might strike us unexpectedly, even if the word can’t be more familiar to us. For example, “watch” and “catch”, both having difficulty around 550, are pretty hard according to human-distributions. The reason for that is they’re having the “tch” letter combination, which you might not classify as English stuff, and also, there’re only a few English words that end with the letter “h”. More combinations like this: “uze”, “wky”, “ltz”, “zes”, “ulc”, “rge”...

1.4 Optimize through Linear Algebra

Given that words in different clusters have different characteristics, and words within the same clusters share similar characteristics. For each cluster, we perform a unique linear transformation aiming to make machine-distributions better “fit” human-distributions. The outcome really surprised us! Matrices have really magically modified our machine prediction in such a way that it is multiple times “closer” to human-distributions.

We’ll show how we improve our distribution model for the word “eerie”, which is an uncommon word with duplicate letters.

1.4.1 Basic Ideas

We first need to accurately define the "closeness", or, how far away are our machine-distributions from the actual human-distributions? As we have regarded distribution as vectors, it won't be hard. For a certain word S, we characterize the "distance" between its machine-distribution $\mathbf{v}(S)$, and its human-distribution $\mathbf{h}(S)$ by:

$$\|\mathbf{v}(S) - \mathbf{h}(S)\|_2^2$$

For a certain cluster, we denote the set of all words in that cluster as $\mathcal{S}$ (the dictionary of this cluster) and define the average distance between all corresponding machine-distributions and human-distributions as:

$$W_0 = \frac{1}{|\mathcal{S}|} \sum_{S \in \mathcal{S}} \|\mathbf{v}(S) - \mathbf{h}(S)\|_2^2$$

What we mean by optimizing by the linear algebra is to find such a matrix T that after multiplying every $\mathbf{v}(S)$ with T and then calculating the average distance again would make the result smaller than that before the optimization. Which is to say, this is what we want:

$$W_{\mathcal{S}}(T) = \frac{1}{|\mathcal{S}|} \sum_{S \in \mathcal{S}} \|\mathbf{v}(S) \cdot T - \mathbf{h}(S)\|_2^2 < \frac{1}{|\mathcal{S}|} \sum_{S \in \mathcal{S}} \|\mathbf{v}(S) - \mathbf{h}(S)\|_2^2 = W_0$$

And we want to minimize $W_S(T)$ with T^* to make our machine-distributions close enough to human-distributions. Formally, we are looking for:

$$T^* := \arg\min_T \{W_{\mathcal{S}}(T)\}$$

Constructing such a transforming matrix T^* seems intuitive. We just need to pick a group of the arbitrary linear basis from $\{\mathbf{v}(S) \mid S \in \mathcal{S}\}$, say $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_7$, and place them as column vectors in matrix A:

$$A^T = \begin{bmatrix} \mathbf{v}_1^T & \mathbf{v}_2^T & \mathbf{v}_3^T & \mathbf{v}_4^T & \mathbf{v}_5^T & \mathbf{v}_6^T & \mathbf{v}_7^T \end{bmatrix}$$

Also, find the corresponding (i.e. $\mathbf{h}_i$ and $\mathbf{v}_i$ are corresponding $\iff \exists S, S \in \mathcal{S}$ s.t. $\mathbf{h}_i = \mathbf{h}(S)$ and $\mathbf{v}_i = \mathbf{v}(S)$) human-distributions $\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_3$ of $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_7$, respectively, and place them in matrix B similarly:

$$B^T = \begin{bmatrix} \mathbf{h}_1^T & \mathbf{h}_2^T & \mathbf{h}_3^T & \mathbf{h}_4^T & \mathbf{h}_5^T & \mathbf{h}_6^T & \mathbf{h}_7^T \end{bmatrix}$$

As A are formed by bases, meaning that row vectors of A are linearly independent, thus, A^{-1} must exist, and T is as follows:

$$T = A^{-1} \cdot B$$

When carefully selecting $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_7$, machine-distributions might get improved (closer to

human-distribution, making $W_S(T)$).

However, simply setting $T = A^{-1}B$ doesn't work well and it might even worsen your machine-distributions (make $W_S(T) > W_0$ sometimes, which is what we don't want). We need to improve the magic T, and encourage it to do a better job.

1.4.2 Reducing Dimensions

Notice that for $\mathbf{v} = (v_1, v_2, v_3, v_4, v_5, v_6, v_7)$, v_1 are close to zero most of the time. So, we decide to remove the first dimensions of vector $\mathbf{v}$ to make our optimization easier, forming a new 6-dimensional vector $\mathbf{v}' = (v_2, v_3, v_4, v_5, v_6, v_7)$. Thus, all of the matrices A, B, and T would turn into 6×6 matrices with redundant information removed.

1.4.3 Rewriting Matrix A

We define that matrix $D = A - B$, and then $A = B + D$. Matrix D would provide us with some freedom to tweak the transformation T manually so that the average distance $W_S(T)$ can be smaller.

If we focus on matrix D itself, then T has no direct relationship with A:

$$T = (B + D)^{-1} \cdot B$$

Take the word "eerie" for example, which we believe lays in either the **blue** or the **purple** cluster (recall the colored clusters in Figure 2). Here, we only consider that 40 words with duplicate letters inside the **blue** and **purple** cluster, because they share similar attributes and difficulties. Here's all of them:

gorge	abbey	knoll	elder	dodge	vivid	shall	lowly	fewer	natal
comma	larva	forgo	canny	delve	gloom	goose	cacao	droll	egret
sever	fluff	madam	wedge	motto	buggy	ruder	gully	booze	foggy
libel	brave	eject	excel	ionic	mummy	swill	trite	cinch	dandy

Table 3: Words believed to have similar attributes with "eerie"

All of these words' machine-distribution can be viewed in 4.1.

1.4.4 Finding Matrix D via Program

Instead of editing matrix D manually, we can use a program that changes D slightly each time and check whether it makes $W_S(T)$ smaller to decide whether or not the changes are worthy.

For a given matrix D, we can regard it as 36 variables, and what we want is that the machine can adjust it properly in such a way that it reaches the local minima for the function $W_S(T)$ in the 36-dimensional space $\mathbb{R}^{36}$.

Metaheuristic such as simulated annealing comes to our mind, which is a heuristical algorithm that is good at looking for the local optimum.

Through simulated annealing, we found the desired matrix D and T^*, where:

$$D = \begin{bmatrix} -5.727 & 51.927 & 6.574 & -20.198 & -11.676 & -20.713 \\ -1.45 & -10.932 & 11.664 & 2.154 & -3.181 & 2.039 \\ 2.945 & -31.309 & 8.986 & 11.281 & -5.067 & 13.221 \\ -1.755 & -15.045 & 13.927 & 1.628 & -0.717 & 2.628 \\ 0.875 & 19.103 & 13.279 & -11.302 & -16.758 & -4.616 \\ 4.01 & -63.327 & 3.686 & 27.529 & 11.219 & 17.418 \end{bmatrix}$$

and most importantly:

$$T^* = \begin{bmatrix} 0.25246474 & 0.6411621 & -0.26101408 & -0.03436244 & 0.32426379 & -0.0840906 \\ 0.04990661 & 0.20029297 & 0.23130413 & 0.1591435 & 0.20678471 & 0.17988505 \\ 0.01137766 & 0.13040194 & 0.39854058 & 0.28850659 & 0.16867407 & -0.0369755 \\ 0.00267197 & 0.03332336 & 0.36919214 & 0.63096927 & 0.05530095 & -0.03515383 \\ 0.02782655 & 0.04299187 & -0.10379562 & 0.0784446 & 0.64083388 & 0.25617677 \\ 0.04550392 & 0.33513107 & 0.24559258 & -0.24550954 & 0.23117789 & 0.46359284 \end{bmatrix}$$

To view it clearly: (The brighter the larger number it is)

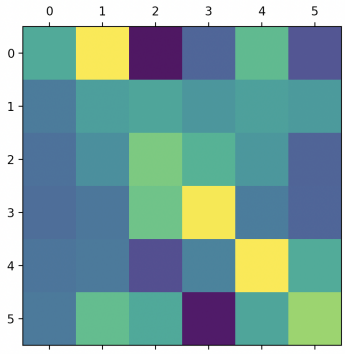

Figure 3: Visualization of matrix T^*

To verify that matrix T^* really helped us optimize machine-distributions, we compared the average distance between distributions before the optimization and after it. The original (i.e. without optimization) $W_0 = 306.704$, after multiplying every $\mathbf{v}(S)$ by T, $W_S(T^*)$ becomes 52.835. The exciting news here is that T^* really makes our machine-distributions much much more accurate (closer to human-distributions).

$$\text{improvment} = \frac{W_0}{W_S(T^*)} = \frac{306.704}{52.835} \approx 6(\text{times})$$

Here's an intuitive graph that shows how much improvement linear transformation provides us. The **red** line is a dataset of $\|\mathbf{v} - \mathbf{h}\|_2^2$ for each word before the optimization, and the **green** line is a dataset of $\|\mathbf{v}T - \mathbf{h}\|_2^2$ after the optimization.

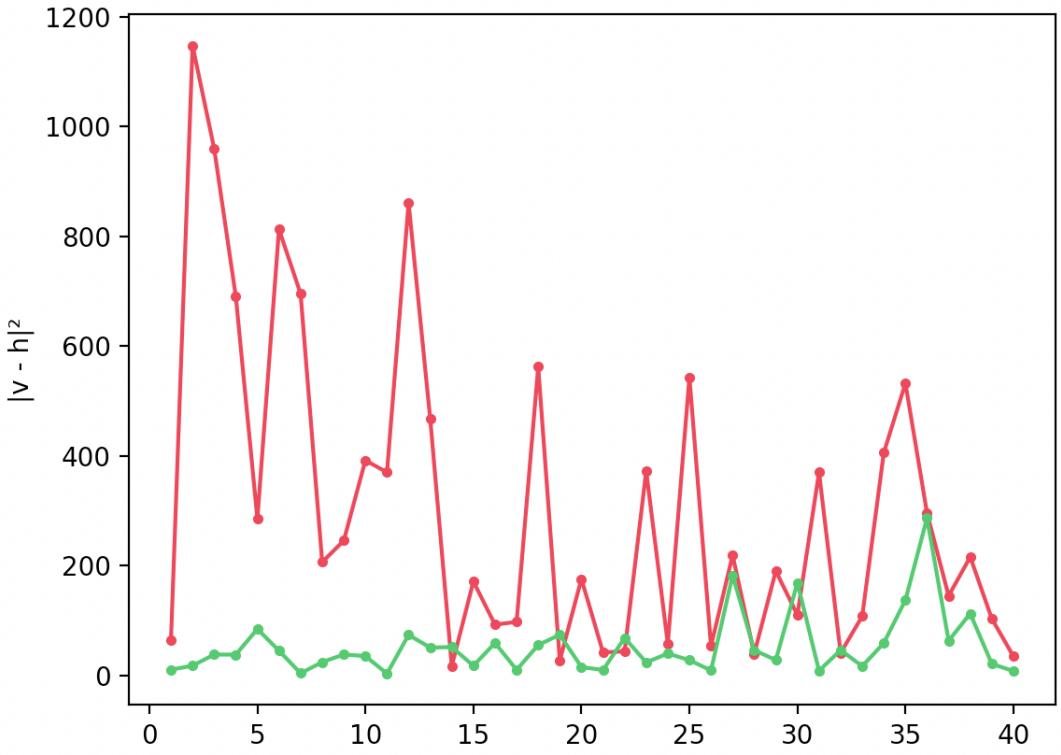

Figure 4: Optimized machine-distributions

Complete dataset can be viewed in 4.2 on each word's human-distributions $\mathbf{h}$, modified machine-distributions $\mathbf{v}$, and respective $\|\mathbf{v} - \mathbf{h}\|_2^2$

Back to the word "eerie", the machine-distribution we obtained is, and the more accurate machine-distribution (obtained after multiplying T) are:

$$\mathbf{v}_0(\text{eerie}) = (2.2 \quad 15.3 \quad 31.6 \quad 34.2 \quad 13.9 \quad 2.8)$$

$$\mathbf{v}(\text{eerie}) = \mathbf{v}_0(\text{eerie})T = (2.2214 \quad 11.2286 \quad 27.4110 \quad 33.4029 \quad 20.5972 \quad 5.0741)$$

By assume that almost no one can figure out "eerie" in their first try, we claim our final estimation for "eerie"s distribution of result is that:

$$(0 \quad 2 \quad 11 \quad 27 \quad 34 \quad 21 \quad 5)$$

And we can calculate the difficulty of "eerie", which is 4.91, with the help of its distribution.

2 Predicting Number of Results

To make a prediction on how many wordle results would be reported on a specific date, we need to refer to the number of results that had been reported throughout 2022. We introduced the daily result counts (i.e. the number of reported results) to the computer. We then established the relationship between dates and result-counts by modeling it via fractional functions. However, such a relationship doesn't necessarily indicate the actual number, and fluctuations between the estimated result-counts and the actual result-counts are unavoidable. However, we still succedd in using this relationship along with the fluctuation to predict the results would be reported on the exact date of March 1st, 2023.

2.1 Processing the Data

When we process the data, we choose to plot the result-counts with respect to weeks. It is because we want to eliminate, or at least minimize, the effect of which day it is. We expected that result-counts would vary from weekdays and weekends. The number of results soared in the first three weeks. A possible and reasonable explanation is that the game wordle is trending around the globe. As we go past the three-week period, we observe a gradual reduction until the end of the curve gradually flattens, indicating that the reduction is closing to 0.

2.2 Approximation through Fractional Function

After plotting the initial graph with the raw data, we observe that the points are too scattered to show a specific overall pattern. Hence, we choose to distribute the 359 data into 7 groups and discard the remaining data. we obtained 51 sets of data by combining the result-counts together week by week. After plotting with the modified dataset, we arrived at a graph that is smoother and successfully, to a great extent, reduced the unnecessary wiggles in the curve.

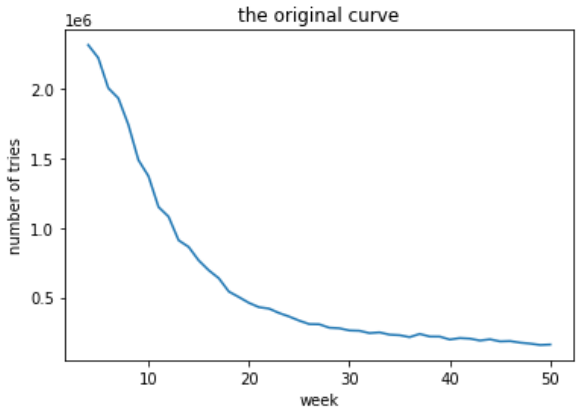

Figure 5: Original curve

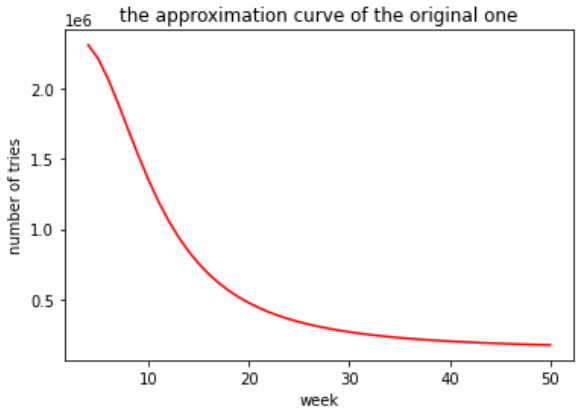

Figure 6: Fitting curve

By observing the curve even further, we noticed that the curve decreases in a constant but slow manner. Therefore, we choose fractional functions to form a proper fit, which is in the form of $f(x) = \frac{a_1x^2+b_1x+c_1}{a_2x^2+b_2x+c_2}$, being able to best describe the trend of the original curve and can, therefore, help with the prediction of the result-counts in the future. Furthermore, when estimating the result-counts for March 1, 2023, we chose to desert data from the first 20 weeks, so that the relatively stable rate of change can guard our accuracy while preventing being affected too much by the unusual changes from the first week to week 20.

With helpful mathematical tools, we were capable of obtaining the formula of the approximated function.

$$f(x) = \frac{-46549x^2 + 2744583x + -9838549}{-0.29840x^2 + 21.65411x + -265.64714}$$

Here's an insightful look into how well the fitting curve works:

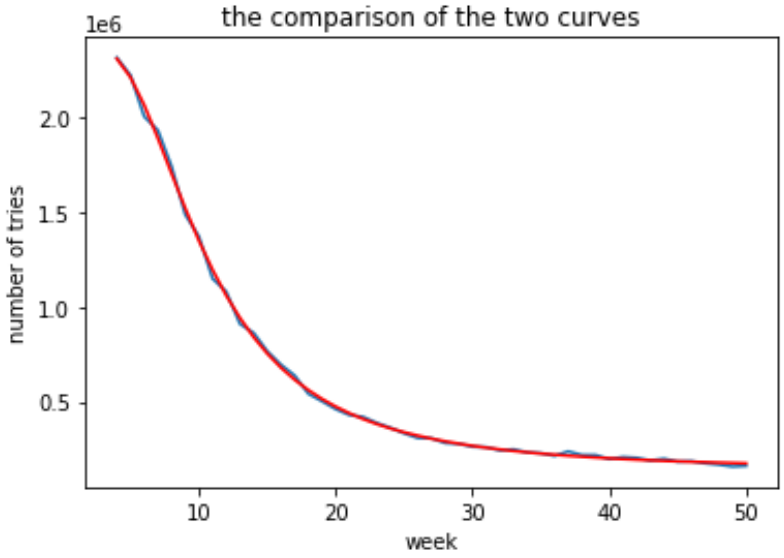

Figure 7: Original curve versus fitting curve

By modeling like this, we can calculate the number of reported results for the 61^{st} week C_{61} (note that March 1, 2023 falls in that week).

$$C_{61} = f(61) \approx 160160$$

In the beginning, we were expecting some noticeable differences of result-counts between Mondays, Tuesdays, ..., and Sundays. To our surprise, the distribution within a week is somehow pretty close to even. We denote C_0 as the estimated number of reported results on March 1, 2023, then we roughly get:

$$C_0 = \frac{f(61)}{7} \approx 22880$$

2.3 Calculating Confidence Interval

However, an interval may be more informative than a single number. To obtain a range for result-counts and their confidence interval, our idea is to compared each day's actual result-counts with the fitting curve's. We denote the reliability of date d as: (where $c(d)$ denotes the actual result-counts of date d)

$$r(d) = \left\lfloor \frac{c(d)}{f(d)} \times 100 \right\rfloor$$

After that, we expect a normal distribution from the histogram of $r(d)$:

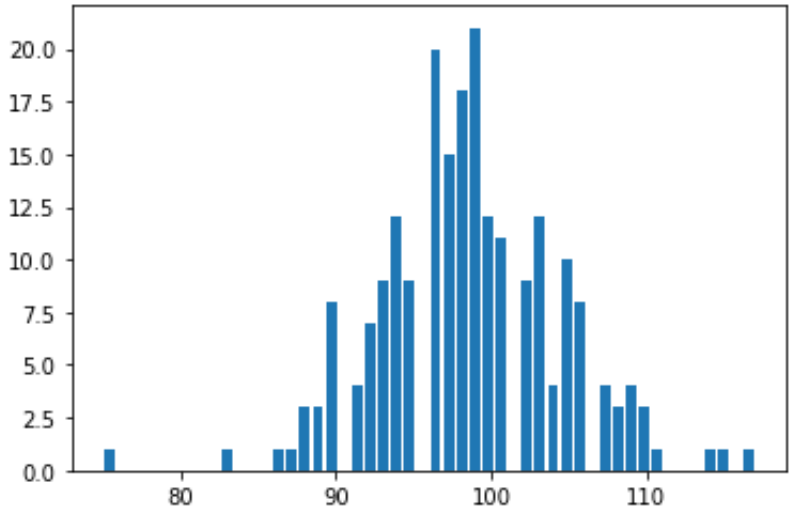

Figure 8: The histogram of $r(d)$

It's rather easy to deal with normal distributions. When $r(d) \in [89, 105]$, 87.2% of the data fits in this range. Thus we can tell with 87.2% confidence that on March 1, 2023, the number of reported results would lay somewhere between:

$$[20935, 23909]$$

3 Letter to the Puzzle Editor of the New York Times

To whom it may concern,

We're pretty glad to have the chance to analyze the results of 359 wordle games. Not only have we constructed a result-counts (the number of results reported) function with respect to time, but we also have built a model that can give a fairly accurate estimation of the distribution of reported results and also assign difficulty to the word fed to it.

Let's first talk about the result-counts versus time function. We notice that the reported result rose sharply after Jan 7, 2022, and drops dramatically after three months. Interpretation such as Wordle was exposed to the public for the first time and was gaining attention are valid and good enough to explain the "steep cliff" in the result-counts versus time graph ("the curve" for short). Once Wordle leaves the center of the public's sight, the curve becomes flat and goes down slowly.

Mathematically, we use a fractional function to fit the curve, and $f(x) = \frac{49860x^2-1088531x+53270141}{0.319^2-2.199x+25.195}$ outcompetes other models and reports that there would be 22880 reported results on March 1, 2023. However, it is very unlikely that this estimation is the exact value that will be reported, and we do some further modeling and believes that it's very likely that the number of reported results would lies in the range [20935, 23909]. But we still doubt that the actual situation would be much more complicated, and we expect a sudden increase at that date because this analysis task brings Wordle back to the public's attention and many research teams will play wordle at least once on March 1, 2023.

We don't logically believe that there would be a connection between the rate of reported hard mode results and the attribute of the solution word of that day. Since the player has to choose the mode before his/her first guess, such a decision would have nothing to do with the solution word's attribute unless the player knows the solution word beforehand, which, indeed, ruins the experience of wordle.

The fun part is building the model which estimates the distribution of reported results and assigns difficulties to the result word. We wrote a program that simulates how a human plays wordle, and by making the program "wordle" the same solution word several times, you would get the so-called "machine-distributions". And we apply different linear transformations on machine-distributions with different characteristics to modify the "machine-distributions" in such a way that they resemble "human-distributions". However, we assume that dates don't influence the distribution of reported results much, and our model estimate that the word "eerie" has a distribution of (0, 2, 11, 27, 34, 21, 5).

We believe that the distribution of reported results will reveal how difficult the solution word is to most people or an average human. Our model of difficulty is completely based on our model of distribution because we think that there doesn't exist a solution word's attribute such that it affects the word's difficulties only but does not affect that word's distribution of results. We basically just calculate the expected tries that humans needed to figure out the solution word, and claim it as the "difficulty". The word "eerie" has a difficulty of 4.91 (which is rather hard), meaning that an average human can

figure it out in around 5 guesses. We applied a clustering algorithm to distributions of reported results and find that there're 5 levels of difficulty in this dataset, and "eerie" are believed to lie in the second hardest cluster.

Through clustering, we also spotted some interesting attributes of the solution word that would affect its distribution and difficulty. We will list them here to be a reference for how to make further wordle solution words interesting.

Familiar to most people If a word is so rare that nearly no one can come it up when playing wordle, then that word is not suitable for wordle. For example, the word "parer" is unfamiliar to most people which leads to a failure rate of 48%.

Duplicate letters If a not-so-common word has duplicate letters in it (i.e. the same letter appears more than once in that word), then it would make the word harder to be figured out. The mechanism behind it is that most people tend to test for un-tried letters instead of considering duplicate letters. Furthermore, some people regard using the same letter in the same guess as not informative. So, using duplicate letters can make a wordle solution word seems mysterious.

Rare consonants When the solution word contains one of the least common consonants in the English alphabet — Q, J, X, Z, W, K, V, or K, players might need extra guess to hit the desired consonant. On the other side, containing R, A, I, S, or E will make the feedback contains more green or yellow blocks.

Words of the same structure Have a close look at "shake" , "share", "shave", "shade", "shame", and "shape", they are all common words with the same "sha?e" structure. Players might be able to figure out this structure after a few guesses, but they won't feel so good after that. Which one is the desired answer?

Rare letter combination Rare words with letter combination of "uze", "wky", "ltz", "zes", "ulc", or "rge" might make players doubt that whether the solution words is an English word or not. However, these words require players to have a better inferential capability and larger vocabulary.

That's our findings about the dataset. Every step we took during this anaylsis process is a great lesson we learned.

Yours,

Team 2322298

4 Data

4.1 Example of Machine-Distributions

```
[ 0.  2.4  14.4  31.5  32.2  15.9  3.6 ]
[ 0.  2.6  33.3  44.2  17.8   2.1  0.  ]
[ 0.  1.2  24.5  47.7  23.4   3.1  0.1 ]
[ 0.  3.9  27.6  36.3  23.7   7.6  0.9 ]
[ 0.  1.2  16.5  41.3  30.2  10.2  0.6 ]
[ 0.  0.2  16.6  50.7  26.9   5.4  0.2 ]
[ 0.  5.4  30.   43.9  18.7   2.   0.  ]
[ 0.  0.6  14.8  33.7  32.6  14.2  4.1 ]
[ 0.  2.   13.3  34.   32.6  14.7  3.4 ]
[ 0.  4.8  26.4  38.5  22.2   7.8  0.3 ]
[ 0.  0.6  22.2  43.9  27.2   5.9  0.2 ]
[ 0.  5.4  24.2  39.9  24.3   5.7  0.5 ]
[ 0.  1.7  16.1  40.   32.6   9.   0.6 ]
[ 0.  0.9  10.8  31.   31.7  18.8  6.8 ]
[ 0.  0.8  18.9  40.9  29.7   8.6  1.1 ]
[ 0.  0.9  17.4  41.7  30.7   9.   0.3 ]
[ 0.  3.7  15.8  33.4  28.5  14.9  3.7 ]
[ 0.  1.3  20.8  38.6  31.8   6.9  0.6 ]
[ 0.  1.1  12.4  34.2  32.6  16.3  3.4 ]
[ 0.  3.2  20.7  38.7  31.8   5.3  0.3 ]
[ 0.  0.8  14.9  32.2  28.1  18.2  5.8 ]
[ 0.  0.    6.   29.5  43.2  19.   2.3 ]
[ 0.  0.3  23.8  44.8  25.5   5.2  0.4 ]
[ 0.  1.3  14.1  35.1  33.3  12.4  3.8 ]
[ 0.  0.8  23.8  47.5  24.7   3.1  0.1 ]
[ 0.  0.4  10.2  33.4  33.6  16.4  6.  ]
[ 0.  0.7   7.1  26.8  36.6  21.7  7.1 ]
[ 0.  0.3   9.9  27.8  28.8  21.6 11.6 ]
[ 0.  0.5  10.7  36.   37.5  13.5  1.8 ]
[ 0.  0.1   8.4  29.2  35.8  20.   6.5 ]
[ 0.  2.4  26.   40.8  26.5   4.3  0.  ]
[ 0.  3.8  18.3  31.   26.7  13.9  6.3 ]
[ 0.  1.   15.1  40.3  37.    6.4  0.2 ]
[ 0.  2.4  26.3  40.4  22.8   7.3  0.8 ]
[ 0.  1.4  21.3  38.   31.    7.7  0.6 ]
[ 0.  0.    5.5  22.8  34.3  26.3 11.1 ]
[ 0.  0.6   7.6  27.9  34.1  23.5  6.3 ]
[ 0.  2.8  15.4  29.7  28.5  17.7  5.9 ]
[ 0.  0.7  11.2  28.8  26.9  18.7 13.7 ]
[ 0.  0.5   7.9  26.1  30.9  21.6 13.  ]
```

4.2 Human-Distributions v.s. Optimized Machine-Distributions v.s. $\|\mathbf{v}-\mathbf{h}\|_2^2$

```
[ 1.  3. 13. 27. 30. 22.  4.], [ 0. 2.2 11.4 26.3 31.9 21.8  5.8], 10.5
[ 1.  2. 13. 29. 31. 20.  3.], [ 0. 2.8 14.7 30.9 29.3 17.5  4. ], 17.8
[ 1.  1. 11. 29. 33. 21.  4.], [ 0. 2.  12.8 32.7 32.6 16.8  2.5], 38.1
[ 1.  3. 13. 24. 30. 24.  5.], [ 0. 2.9 14.1 28.  30.  19.4  4.8], 37.9
[ 1.  3. 15. 29. 27. 19.  7.], [ 0. 1.8 11.1 30.2 34.2 19.1  3.1], 84.6
[ 1.  2. 10. 29. 33. 21.  4.], [ 0. 1.5 11.2 33.4 34.6 17.   1.6], 45.1
[ 0.  3. 17. 30. 28. 17.  4.], [ 0. 3.3 15.9 29.7 29.2 17.6  3.1], 4.0
[ 0.  2.  9. 26. 32. 24.  8.], [ 0. 1.7 10.8 28.2 32.7 20.7  5.7], 24.3
[ 0.  2. 10. 24. 32. 26.  6.], [ 0. 2.  11.2 27.4 32.7 21.1  5.1], 38.2
[ 0.  2. 14. 32. 32. 17.  3.], [ 0. 3.1 14.5 27.6 29.6 19.8  4.2], 35.3
[ 0.  2. 13. 33. 33. 17.  3.], [ 0. 1.8 11.7 31.9 33.7 17.5  2.9], 3.4
[ 0.  2. 10. 25. 35. 23.  4.], [ 0. 3.1 14.7 28.5 30.8 18.5  3.2], 73.9
[ 0.  1.  9. 26. 37. 23.  3.], [ 0. 1.8 11.2 30.4 35.1 18.3  2.7], 50.7
[ 0.  2. 10. 30. 34. 20.  4.], [ 0. 1.8 10.9 26.  30.4 23.1  7.5], 51.7
[ 0.  2. 14. 32. 33. 16.  2.], [ 0. 1.8 11.3 30.8 33.9 18.4  3.4], 17.6
[ 0.  2. 14. 35. 35. 13.  1.], [ 0. 1.7 11.  30.8 34.7 18.4  2.8], 59.3
[ 0.  2. 12. 28. 32. 21.  5.], [ 0. 2.6 12.7 25.8 30.2 22.   5.8], 10.2
[ 0.  1.  9. 27. 36. 23.  4.], [ 0. 1.9 11.5 31.  34.8 17.5  3.1], 55.5
[ 0.  2. 16. 31. 31. 17.  3.], [ 0. 1.8 10.5 27.3 32.8 21.7  5.4], 74.2
[ 0.  3. 14. 33. 33. 15.  2.], [ 0. 2.3 12.6 30.6 34.7 17.   2.4], 15.3
[ 0.  2. 13. 27. 29. 21.  7.], [ 0. 2.  11.3 25.9 29.3 23.3  7.7], 9.9
[ 0.  0.  4. 25. 44. 23.  4.], [ 0. 1.1  8.  27.6 37.6 21.3  4.4], 68.4
[ 0.  3. 13. 35. 34. 14.  2.], [ 0. 1.8 12.  32.2 33.1 17.4  3.2], 23.7
[ 0.  2. 11. 32. 37. 17.  2.], [ 0. 1.8 11.1 28.8 33.3 19.9  4.8], 40.0
[ 0.  1. 11. 36. 36. 14.  1.], [ 0. 1.9 12.4 33.  33.2 16.5  2.4], 27.8
[ 0.  1.  9. 29. 34. 22.  5.], [ 0. 1.6 10.4 27.7 32.2 21.6  6.3], 9.2
[ 0.  2. 16. 33. 29. 16.  4.], [ 0. 1.6  9.8 25.1 31.8 23.7  7.7], 182.4
[ 0.  1.  9. 27. 31. 25.  7.], [ 0. 1.9 11.5 24.5 26.6 24.9 10.6], 46.0
[ 0.  1.  7. 27. 38. 23.  4.], [ 0. 1.4  9.5 29.5 36.3 19.5  3.5], 28.1
[ 0.  2. 13. 35. 32. 15.  3.], [ 0. 1.5  9.7 26.2 32.3 22.9  7.3], 168.8
[ 0.  3. 15. 32. 32. 16.  2.], [ 0. 2.4 13.1 30.9 32.8 17.2  3.1], 8.5
[ 1.  6. 17. 27. 27. 18.  5.], [ 0. 2.8 13.7 25.5 28.1 22.   7.3], 46.3
[ 0.  2. 11. 35. 36. 14.  2.], [ 0. 1.5 10.4 32.3 37.7 16.4  1.5], 16.8
[ 0.  1. 13. 34. 34. 15.  2.], [ 0. 2.5 13.4 29.4 30.5 19.1  4.4], 59.1
[ 0.  2. 13. 25. 28. 21. 11.], [ 0. 2.  11.6 30.4 34.3 18.   3.4], 137.3
[ 0.  1.  4. 14. 27. 37. 18.], [ 0. 1.6 10.  23.  28.4 26.2 10.8], 286.6
[ 1.  1.  8. 19. 31. 30. 10.], [ 0. 1.6  9.8 24.4 31.  24.8  8. ], 63.2
[ 0.  4. 14. 22. 22. 23. 15.], [ 0. 2.4 12.4 24.8 28.8 23.3  7.7], 112.7
[ 0.  2. 15. 24. 22. 25. 13.], [ 0. 2.1 12.7 25.2 25.1 24.  11. ], 21.1
[ 0.  2. 11. 23. 29. 24. 11.], [ 0. 1.9 11.6 24.4 26.7 24.7 10.8], 8.0
```

5 References

1. The Wordle Game [New York Times] — https://www.nytimes.com/games/wordle/index.html
2. Solving Wordle using information theory [3Blue1Brown] — https://www.youtube.com/watch?v=v68zYyaEmEA
3. WordFrequencyData [Wolfram] — https://reference.wolfram.com/language/ref/WordFrequenc
html
4. Letter Frequency [Wikipedia] — https://en.wikipedia.org/wiki/Letter_frequency
5. K-Means Clustering [Wikipedia] — https://en.wikipedia.org/wiki/K-means_clustering
6. Simulated Annealing [Wikipedia] — https://en.wikipedia.org/wiki/Simulated_annealing