

# Finding an Optimal Wordle Strategy Using Mathematics

Page Count: 20

## **I. Introduction**

- Wordle Game Rules
- Game Objective
- Wordle Strategies
- Aim

## **II. Methodology**

- General Terminology, Variables, and Notation
  - Terminology
  - Variables
  - Notation
- Set-Up
- Exploration Structure

## **III. Strategy 1: Random**

- Explanation
- Data Collection & Processing
- Evaluation

## **IV. Strategy 2: Entropy**

- Explanation
- Data Collection & Processing
- Evaluation

## **V. Strategy 3: Expected Score**

- Explanation
- Data Collection & Processing
- Evaluation

## **VI. Conclusion**

- Results
- Methodological Strengths & Limitations
- Extensions
- Takeaways

## **References**

### **Appendix A: Strategy 1**

### **Appendix B: Strategy 2**

### **Appendix C: Strategy 3**

# I. Introduction

“Did you do today’s Wordle yet?”

“Yeah, and I got it in 3!”

Ever since the word game Wordle started trending on the internet in January 2022, I hear and engage in these kinds of conversations all the time. After I was first introduced to Wordle in early February 2022, I have played it religiously, eagerly waiting for 12 a.m. every day to play the newest game. My friends and classmates all play Wordle and we often compare results, strategies, and performance statistics.

## *Wordle Game Rules*

Every day, a new Wordle **game** is available: a mystery 5-letter word is selected from the **answer list** as the answer, and the player has a maximum of 6 attempts to guess the correct word, with each guess required to be a valid 5-letter word from the **guess list**. For each guess, the player receives **feedback** in the form of coloured character tiles (see *Figure 1* for an example of a game):

- a tile turns **green** if the letter is in the word and in the correct spot
- a tile turns **yellow** if the letter is in the word but in the wrong spot
- a tile turns **gray** if the letter is not in the word in any spot

A	R	O	S	E
A	D	E	P	T
A	H	E	A	D

*Figure 1:* Using the guess sequence AROSE, ADEPT, AHEAD when the answer is AHEAD yields the display shown.

In some words from the answer list and guess list, there are repeated letters. In these cases, feedback is given on a letter-by-letter basis, prioritizing **green** tiles over **yellow** tiles and moving from left to right. For example, if the correct answer was RUPEE, the guess EERIE would be displayed as: **EERIE**.

## ***Game Objective***

To **win**, the player must solve the Wordle within 6 tries. However, for most people (myself included), the true goal is to not only win Wordle, but to do so in as few guesses as possible—the fewer the number of guesses taken, the greater the bragging rights!

## ***Wordle Strategies***

So, how do players typically go about solving Wordle? Everyone has their own Wordle strategy and their own favourite way to start a game: some people change their first guess from day to day, while others swear by the same one; some people start by using as many vowels as possible, while others try to maximize the number of common letters in their first guess.

Personally, I always start with STARE, as I find it has lots of common letters whose presence or absence can help me narrow down my search. For each guess after the first one, I almost always implement the feedback received on the previous guess in my next guess and simply guess whatever word comes to mind. While this approach is not very systematic, my strategy seems to be working fairly well for me—in the past 25 days, I have won the Wordle every day, with a mean number of guesses taken of 3.76 (see *Sample calculations 2* and *9* as an example of how this mean was calculated).

## ***Aim***

The more I've played Wordle, the more I've wondered whether there exists an “optimal” strategy. I figured there must be a more rigorous, systematic, and mathematically-sound approach than simply picking “whatever word comes to mind”. Through this exploration, my aim is to use mathematics to develop and analyze several Wordle strategies and hence find an optimal Wordle strategy that performs better than I do (100% win rate and 3.76 mean number of guesses). Given that “optimal” is a subjective term, I will analyze the quality of a strategy using two metrics (sorted below by priority):


1. Probability of winning,  $P(win)$ : This metric denotes what portion of all 2315 possible games are won by the strategy (i.e.  $\leq 6$  guesses are taken), with a higher win rate being more favourable.

2. Expected (i.e. mean) number of guesses,  $E(G)$ : This metric will reveal how well the strategy solves Wordle, with a lower mean being more favourable.

## II. Methodology

### *General Terminology, Variables, and Notation*

**Table 1:** Terminology, variables, and notation that will be used throughout this exploration. Terminology, variables, and notation specific to each strategy will be introduced as they appear in the exploration.

<p><b><u>Terminology</u></b></p> <ul style="list-style-type: none"> <li>➤ <b>Game:</b> A game consists of a correct answer selected from the answer list and a sequence of guesses taken in an attempt to find the answer. A game is <b>won</b> if the number of required guesses to solve the Wordle does not exceed 6.</li> <li>➤ <b>Answer list:</b> The official list of 2315 words (Chow, 2022) from which the mystery word is chosen in a game of Wordle.</li> <li>➤ <b>Guess list:</b> The official list of 12972 words (2315 answer words and 10657 additional accepted guesses) (Chow, 2022)</li> <li>➤ <b>Feedback:</b> The letter-by-letter colours displayed for a given guess.</li> <li>➤ <b>Consideration list:</b> The list of words deemed possibly correct at a given point in a game by the algorithm. The consideration list is filtered as feedback is received.</li> </ul>
<p><b><u>Variables</u></b></p> <ul style="list-style-type: none"> <li>➤ <math>G</math> is the random variable denoting the set of all possible number of guesses, <math>g</math>, needed to solve a Wordle: <math>g \in G</math></li> <li>➤ <math>g</math> is the specific number of guesses needed to solve a specific Wordle, where <math>\{g \in \mathbb{Z} \mid g \geq 1\}</math>. If <math>g &gt; 6</math>, the game is lost.</li> </ul>
<p><b><u>Notation</u></b></p> <ul style="list-style-type: none"> <li>➤ Feedback will be expressed as a 5-digit ternary number (i.e. consisting of 0s, 1s, and 2s). 0 represents gray, 1 represents yellow, and 2 represents green. For instance, the feedback  would be displayed as 10102.</li> <li>➤ <math>P(G = g)</math> represents the probability that a strategy wins wordle in <math>g</math> guesses.</li> </ul>

### *Set-Up*

In order to analyze different Wordle strategies, I coded several Wordle programs in C++, as large quantities of data need to be filtered through and processed to obtain results. The basic Wordle program I

created returns feedback on a given guess based on a comparison against the answer, filters the consideration list based on the returned feedback, counts the number of guesses taken, and determines when a Wordle is solved (i.e. the guess is equal to the answer). The primary difference between successive Wordle-solving strategies is the way in which it determines the most optimal next guess.

Programs were designed to collect data and to simulate the different strategies over a large number of games. The answer list of all 2315 Wordle words and the guess list of all 12972 possible Wordle guesses were both retrieved from Cyrus Freshman's GitHub (2022) and stored in each program. I have chosen to make the initial consideration list equal to the guess list (instead of only from the answer list), as I found that it would be more universal and similar to the actual gameplay—after all, most players play Wordle without knowing which words actually belong to the answer list.

To test each strategy, I want to consider all possible Wordle games, so I will iterate through the answer list 2315 times, setting each word as the answer and testing how many guesses the strategy in question requires to solve that specific puzzle. I will allow the computer to use more than 6 guesses in a game for data collection purposes, but such a game will be considered a loss and will lower the win rate.

In this exploration, all data will be presented to 3 significant figures, as this is the convention and will offer enough precision for accurate analysis. All numbers are pure numbers (as no measurements yielding uncertainty were taken) and will be inputted into calculations as such, but rounded values will be displayed. Tools used in this exploration include Desmos to produce graphs and Visual Studio Code to develop programs that implement the strategies.

## ***Exploration Structure***

This exploration will analyze a progression of three Wordle strategies, with the analysis of each strategy being subdivided into three subsections: ***Explanation*** (developing and describing the strategy), ***Data Collection & Processing*** (testing the strategy and computing  $P(win)$  and  $E(G)$  for that strategy), and ***Evaluation*** (discussing results, methodological limitations, and potential improvements to the strategy).

### III. Strategy 1: Random

#### *Explanation*

One of the advantages of playing Wordle as a computer (instead of as a human) is that it is able to quickly and comprehensively update the consideration list based on the feedback after each guess. The first strategy I came up with uses this fact and combines it with luck, or randomness: a word is randomly selected from the 12972 possible guesses as the first guess, the consideration list is filtered based on the feedback received from this first guess, a word is randomly selected from the updated consideration list, and so on until the correct answer is guessed (see **Figure A1** in **Appendix A** for a sample run). This strategy solely focuses on solving the puzzle—it is always trying to guess the correct answer.

#### *Data Collection & Processing*

Because Strategy 1 fundamentally relies on randomness, I decided to repeat the experiment 5 times and calculate the mean frequencies for each value of  $g$  using this data (see **Sample calculation 1**). The full table of raw data is in **Table A1** in **Appendix A**.

**Sample calculation 1:** Finding the mean frequency (from 5 trials) of solving the Wordle in 3 guesses,  $\overline{f_3}$ .

$$\begin{aligned}\overline{f_3} &= \frac{\sum_{i=1}^5 f_{3,i}}{5} \\ &= \frac{309+253+225+231+285}{5} \\ &= \frac{1303}{5} \\ &\approx 261\end{aligned}$$

I then converted the averaged frequencies into probabilities to better analyze the probability that a strategy can solve the Wordle in  $G = g$  guesses.

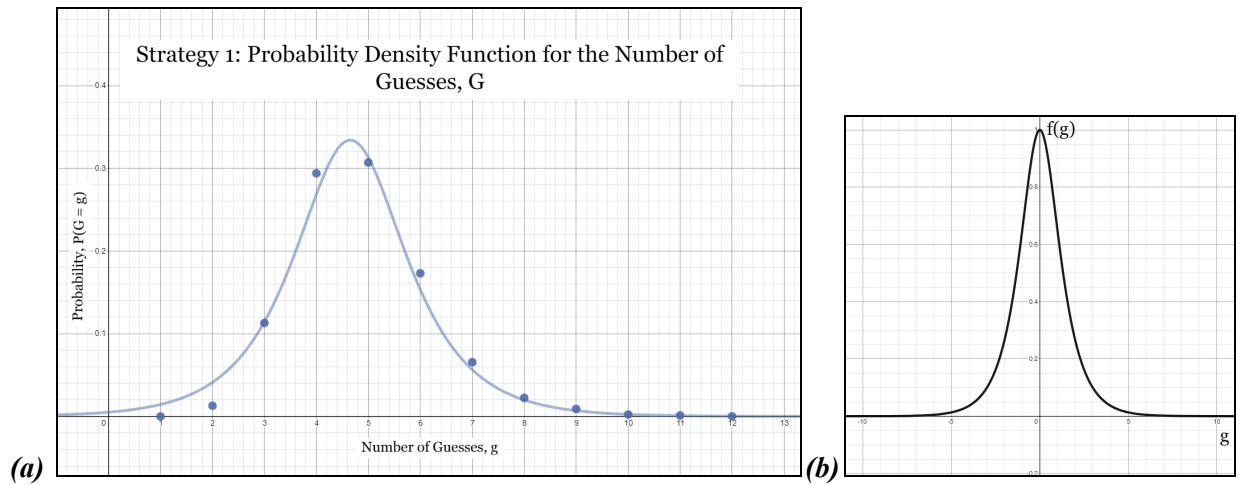
**Sample calculation 2:** Finding the probability of solving the Wordle with 3 guesses,  $P(G = 3)$ , across 2315 total games.

$$\begin{aligned}P(G = 3) &= \frac{\overline{f_3}}{2315} \\ &= \frac{\frac{1303}{5}}{2315} \\ &\approx 0.113\end{aligned}$$

**Table 2:** The probability distribution of  $G$  across 5 trials of 2315 games for Strategy 1.

$g$	1	2	3	4	5	6
$P(G = g)$	0	0.0129	0.113	0.294	0.307	0.173
$g$	7	8	9	10	11	12
$P(G = g)$	0.0656	0.0224	0.00898	0.00233	0.00121	0.0000864

Using this processed data, I created a probability density graph (see **Figure 2(a)** below). Since Strategy 1 is based on randomness, I decided against directly using the data I collected from the 5 trials I conducted to calculate the win rate and the mean number of guesses. Instead, I thought that modeling my results as a continuous probability density function would yield results that are more generally representative of what the results may have been if I were to conduct infinitely many trials. With such a model, I could then integrate the function to find the win rate and the mean. After doing some research, I discovered the that hyperbolic secant function (Glen, 2021),  $\text{sech}(x) = \frac{2}{e^x + e^{-x}}$  (see **Figure 2(b)** below), fit the bell shape of my data well and was suitable for integration and differentiation.

**Figure 2:** (a) Probability density function for Strategy 1. (b) Parent function,  $f(g) = \frac{2}{e^g + e^{-g}}$ .

The curve shown in **Figure 2(a)** is transformed from the parent function: specifically, the equation of the probability density function can be expressed as

$$f(g) = \frac{2a}{e^{b(g-h)} + e^{-b(g-h)}} + k \quad (\text{Eq. 1}), \text{ where } a \text{ is the vertical stretch factor, } b \text{ is the horizontal}$$

compression factor,  $h$  is the horizontal translation, and  $k$  is the vertical translation.

Since the transformed and parent graph both have horizontal asymptotes of 0 (i.e.  $\lim_{g \rightarrow \pm\infty} f(g) = 0$ ),

$k = 0$ . I then adjusted the values of  $a$ ,  $b$ , and  $h$  to fit the data, and found that:  $a \approx 0.335$ ,  $b \approx 1.05$ , and

$h \approx 4.65$ . Substituting these values into **Eq. 1**:

$$f(g) = \frac{2(0.335)}{e^{1.05(g-4.65)} + e^{-1.05(g-4.65)}}$$

$$= \frac{0.67}{e^{1.05(g-4.65)} + e^{-1.05(g-4.65)}}$$

To verify that  $f(g)$  is a valid probability density function, two criteria must be met: ①  $0 \leq f(g) \leq 1$  for

all values of  $g$  and ②  $\lim_{m \rightarrow \infty} \int_{-m}^m f(g) dg = 1$ . Differentiation and integration are needed to show that these

criteria are met, respectively. I will first show the general form of the derivative and the indefinite integral

using **Equation 1**, setting  $k = 0$ , then substituting in values of  $a$ ,  $b$ , and  $h$ .

①: The range of  $f(g)$  is from its horizontal asymptotes of 0 to its maximum (its axis of horizontal symmetry). To find the maximum, I will set the derivative of the function equal to 0 and solve for  $g$ .

**Sample calculation 3:** General form of the derivative of  $f(g) = \frac{2a}{e^{b(g-h)} + e^{-b(g-h)}}$ , using power rule and chain rule. The derivative was then set to 0 to isolate for  $g$  at the maximum.

Differentiation Process		Explanations
$f(g) = 2a(e^{b(g-h)} + e^{-b(g-h)})^{-1}$ $f'(g) = -2a(e^{b(g-h)} + e^{-b(g-h)})^{-2}(b \cdot e^{b(g-h)} - b \cdot e^{-b(g-h)})$ $0 = \frac{-2ab(e^{b(g-h)} - e^{-b(g-h)})}{(e^{b(g-h)} + e^{-b(g-h)})^2}$ $0 = -2ab(e^{b(g-h)} - e^{-b(g-h)})$ $0 = e^{b(g-h)} - e^{-b(g-h)}$ $0 = e^{-b(g-h)}(e^{2b(g-h)} - 1)$		Applying power and chain rule  Numerator must equal 0  Dividing both sides of the equation by $-2ab$ (constant) Factoring out $e^{-b(g-h)}$
$e^{-b(g-h)} = 0$ $-b(g-h) = \ln(0)$ $\therefore$ No solutions.	$e^{2b(g-h)} - 1 = 0$ $e^{2b(g-h)} = 1$ $2b(g-h) = 0$ $g-h = 0$ $g = h$ <b>(Eq. 2)</b>	Setting each factor equal to 0 and solving, if possible.



Therefore, the maximum of a transformed hyperbolic secant curve occurs at  $g = h$  (Eq. 2). For the curve in **Figure 2(a)**, where  $h \approx 4.65$ , the maximum is thus at  $g \approx 4.65$ . Thus,  $f(4.65) \approx 0.335$ , meaning the maximum of  $f(g)$  satisfies  $0 \leq f(g) \leq 1$ .

$\therefore 0 \leq f(g) \leq 1$  is satisfied for all values of  $g$ .

②: Similarly, to verify that  $\lim_{m \rightarrow \infty} \int_{-m}^m f(g) dg = 1$ , I first found the general form of the indefinite integral.

**Sample calculation 4:** General form of the indefinite integral of  $f(g) = \frac{2a}{e^{b(g-h)} + e^{-b(g-h)}}$ , using integration by substitution twice (see right column), as shown below.

Integration Process	Substitutions
$\int \frac{2a}{e^{b(g-h)} + e^{-b(g-h)}} dg = 2a \int \frac{1}{e^u + e^{-u}} \cdot \frac{1}{b} du$ $= \frac{2a}{b} \int \frac{1}{e^u + e^{-u}} \cdot \frac{e^u}{e^u} du$ $= \frac{2a}{b} \int \frac{e^u}{(e^u)^2 + 1} du$ $= \frac{2a}{b} \int \frac{v}{v^2 + 1} \cdot \frac{1}{v} dv$ $= \frac{2a}{b} \int \frac{1}{v^2 + 1} dv$ $= \frac{2a}{b} \arctan(v) + c$ $= \frac{2a}{b} \arctan(e^u) + c$ $= \frac{2a}{b} \arctan(e^{b(g-h)}) + c \quad (\text{Eq. 3})$	$u = b(g - h)$ $du = b dg$ $dg = \frac{1}{b} du$ $v = e^u$ $dv = e^u du$ $\frac{1}{e^u} dv = du$ $\therefore \frac{1}{v} dv = du$

**Sample calculation 5:** Substituting the above expression to evaluate the definite integral  $\lim_{m \rightarrow \infty} \int_{-m}^m f(g) dg$ .

Note that  $\lim_{m \rightarrow \infty} \arctan(m) = \frac{\pi}{2}$  and  $\lim_{m \rightarrow \infty} \frac{1}{m} = 0$  were also used to evaluate the integral.

$$\begin{aligned}
 \lim_{m \rightarrow \infty} \int_{-m}^m \frac{2(0.335)}{e^{1.05(g-4.65)} + e^{-1.05(g-4.65)}} dg &= \lim_{m \rightarrow \infty} \left[ \frac{2(0.335)}{1.05} \arctan(e^{1.05(g-4.65)}) \right]_{-m}^m \\
 &= \frac{0.67}{1.05} (\lim_{m \rightarrow \infty} \arctan(e^{1.05(m-4.65)}) - \lim_{m \rightarrow \infty} \arctan(e^{1.05(-m-4.65)})) \\
 &= \frac{0.67}{1.05} (\lim_{m \rightarrow \infty} \arctan(m) - \lim_{m \rightarrow \infty} \arctan(\frac{1}{m})) \\
 &= \frac{0.67}{1.05} (\frac{\pi}{2} - \arctan(0)) \\
 &= \frac{0.67\pi}{2.10} \\
 &\approx 1.0023 \\
 &\approx 1.00
 \end{aligned}$$

$\therefore f(g) = \frac{2(0.335)}{e^{1.05(g-4.65)} + e^{-1.05(g-4.65)}}$  is a valid probability density function.

The probability of winning with Strategy 1 is thus equal to  $\lim_{m \rightarrow \infty} \int_{-m}^{6.5} f(g) dg$ . To account for the modeling of a discrete probability distribution as a continuous function, the upper bound is extended to 6.5 (by a half interval).

**Sample calculation 6:** Solving for  $P(G < 6.5)$  using *Eq. 3* and following a similar process as in *Sample calculation 5*.

$$\begin{aligned}
 P(G < 6.5) &= \lim_{m \rightarrow \infty} \int_{-m}^{6.5} \frac{2(0.335)}{e^{1.05(g-4.65)} + e^{-1.05(g-4.65)}} dg \\
 &= \frac{0.67}{1.05} \lim_{m \rightarrow \infty} \left[ \arctan(e^{1.05(g-4.65)}) \right]_{-m}^{6.5} \\
 &= \frac{0.67}{1.05} (\arctan(e^{1.05(6.5)-4.8825}) - \lim_{m \rightarrow \infty} \arctan(e^{1.05(-m)-4.8825})) \\
 &= \frac{0.67}{1.05} (\arctan(e^{1.9425}) - \lim_{m \rightarrow \infty} \arctan(\frac{1}{m})) \\
 &= \frac{0.67}{1.05} \arctan(e^{1.9425}) \\
 &\approx 0.911
 \end{aligned}$$

$\therefore P(\text{win}) \approx 0.911$

Since a hyperbolic secant function is symmetric about maximum (Wolfram Research, n.d.), the mean of the probability distribution, or the expected number of guesses, is equal to the  $g$ -value of its maximum, which can be found using *Sample calculation 3*.

$\therefore E(G) = 4.65$

## Evaluation

### Results

Using Strategy 1, the probability of winning is 0.911 and the expected number of guesses is 4.65.

Although this strategy wins on average, there is a significant portion (nearly 10%) of games that are lost, which is not optimal. This can be attributed to the fact that Strategy 1 is entirely dependent on the quality of the guesses made in each round, which itself is entirely random. Out of pure luck, there are some good games (e.g. there exists a non-zero probability of  $G = 2$ ), but there are also many very poorly played games. For this reason, there is a very large range of outcomes and the algorithm is highly inconsistent.

### Analysis of Methodology

Averaging data across several trials and hence creating a model of the probability distribution likely improved the accuracy of my results, as the results from the trials were inconsistent, while  $f(g)$  offers a more generalized view of the guess distribution. Even though there is a strong correlation between the proposed model and the data I collected, there still exists some discrepancies—the model is not perfect. Despite this fact somewhat lowering the reliability of my results, I feel that the advantages of using a continuous probability density function to model the results of Strategy 1 outweigh the drawbacks, since it better simulates what the results of my experiment may have been if I had conducted infinite trials.

### Improvements to Strategy

To improve this strategy, the selection process for each next guess must be intentional, rather than random. Ideally, each next guess reduces the size of the consideration list as much as possible—that is, it provides as much information as possible about the answer. I also realized that always guessing words from the consideration list is not optimal: for example, if the consideration list contains {BATCH, CATCH, LATCH, MATCH, PATCH, WATCH}, guessing any word from this list only eliminates one option each time, meaning it could take up to 6 more guesses to solve the puzzle. Instead, guessing a word like CLAMP is a better choice, since it will either cut the consideration list to {BATCH, WATCH}, or narrow the list down to a single possible answer. Thus, instead of solving the puzzle by constantly guessing from the consideration list, my next strategy will focus on gaining as much information as possible with each guess, and hence narrowing down the consideration list as rapidly as possible.

## IV. Strategy 2: Entropy

### *Explanation*

Intuitively, designing a strategy based on maximizing **information** gained makes sense. In this context, **information** refers to the extent to which a given guess is able to reduce the size of the consideration list. In search of a way to quantify this variable, I stumbled upon a video by Grant Sanderson (2022), who brought up the notion of **entropy** in **information theory**.

**Information theory** is a field within mathematics that studies the quantification of “concepts, parameters and rules governing the transmission of messages through communication systems,” (Martignon, 2001). The basic unit of information is the **bit**, which represents the amount of information required to specify one of two possible values (in the case of a computer, 0 or 1). In Wordle, every time a guess is able to cut the size of the consideration list in  $\frac{1}{2}$ , 1 bit of information is obtained. Thus, the amount of information obtainable,  $I$  (bits), from a guess & feedback pair that has a probability  $p$  of matching with another word in the consideration list (i.e. the list is cut down to  $p$  of its previous size) is:

$$\begin{aligned} \left(\frac{1}{2}\right)^I &= p \\ I &= \log_{\frac{1}{2}}(p) \\ I &= -\log_2(p) \end{aligned} \quad (\text{Eq. 4})$$

I coded this into my program to determine the  $I$  values for each starting guess and possible feedback pair (see **Figure B1** in **Appendix B** for an example using STARE, my personal favourite starting guess!).

With each guess & feedback pair, there is a tradeoff: if it has few matches with other words in the consideration list, it collects a large amount of information, but it also has a low probability of occurring; and vice-versa. To better compare how much information is expected to be obtained from a guess, I needed to incorporate the concept of **entropy**,  $H$ , which is a quantity that is equal to the expected amount of information obtainable,  $E(I)$ , through an event and the probability of this event occurring (Brownlee, 2020).

$$\begin{aligned} H &= E(I) \\ &= \sum_{i=1}^N p \cdot I \end{aligned}$$

Substituting in **Eq. 4**:

$$H = -\sum_{i=1}^N p \cdot \log_2 p \quad (\text{Eq. 5})$$

**Sample calculation 7:** Calculating the entropy (expected information obtained) of STARE using *Eq. 5* and from data in *Figure 1* of **Appendix B**.

$$\begin{aligned}
 H &= - \sum_{i=1}^N p \cdot \log_2 p \\
 &= - ((0.0661) \log_2(0.0661) + (0.0638) \log_2(0.0638) + (0.0626) \log_2(0.0626) + \dots) \\
 &\approx 5.84 \text{ bits}
 \end{aligned}$$

The more evenly spread out the probability distribution across all possible feedback, the greater the entropy (see *Table B2* in **Appendix B** for an example). In Strategy 2, as the first guess, I will always choose the word that yields the greatest entropy—the one that is expected to shrink the consideration list by the greatest factor. After I iterated through all 12972 possible first guesses and calculated each entropy value (see *Table B3* in **Appendix B**), the “optimal” first guess is **TARES**, with an entropy of approximately 6.19 bits! In comparison, my usual starting word, STARE, only ranks 70<sup>th</sup> on this list.

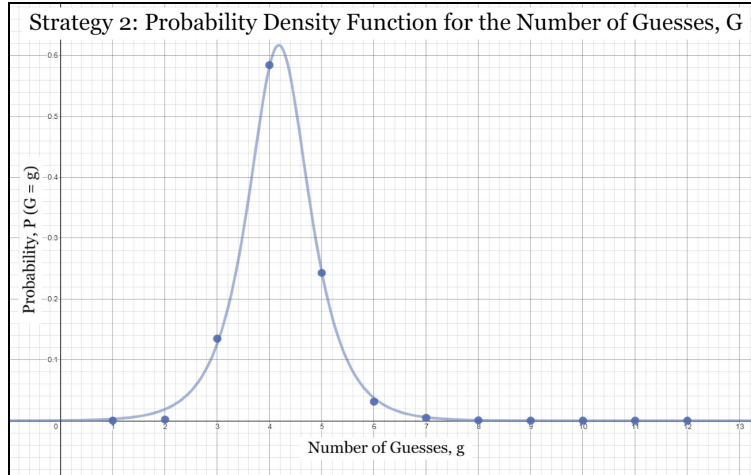
So, Strategy 2 will start every game with TARES as the first guess. I then needed a way to select each successive guess. I realized I could continue using entropy to calculate each next guess: after filtering the consideration list using feedback from the previous guess, each next guess will iterate through the 12972 words in the guess list to find which word will cut down the remaining consideration list to the greatest extent—that is, which guess will yield the greatest entropy within the scope of the consideration list. A sample run is shown in *Figure B2* in **Appendix B**.

## Data Collection & Processing

Using Strategy 2, every answer has a definite (i.e. not random) sequence of guesses: at each guess, there exists a specific word that will yield the greatest entropy for that configuration, which will be chosen as the next guess. For this reason, only 1 “trial” is needed to collect data.

**Table 3:** The probability distribution of  $G$  across all 2315 games for Strategy 2. For better comparison with Strategy 1, probabilities of 0 for  $P(9 \leq G \leq 12)$  are also shown.

$g$	1	2	3	4	5	6
$P(G = g)$	0	0.00173	0.135	0.584	0.243	0.0311
$g$	7	8	9	10	11	12
$P(G = g)$	0.00475	0.000432	0	0	0	0



**Figure 3:** Probability density function for the number of guesses using Strategy 2. Following a similar process to how I curve-fitted **Figure 2(a)**, this distribution has been curve-fitted with a transformed

$$\text{hyperbolic secant function: } f(g) = \frac{1.23}{e^{1.92(x-4.18)} + e^{-1.92(x-4.18)}} \cdot$$

**Sample calculation 8:** The probability of winning Wordle using Strategy 2. Using the model to perform this calculation is not necessary, as there exists a definite probability of winning or losing. Approximately equal symbols are used where substitutions involve rounded values.

$$\begin{aligned} P(\text{win}) &= P(G \leq 6) \\ &= \sum_{g=1}^6 P(G = g) \\ &\approx 0 + 0.00173 + 0.135 + 0.584 + 0.243 + 0.0311 \\ &\approx 0.995 \end{aligned}$$

**Sample calculation 9:** The expected number of guesses using Strategy 2.

$$\begin{aligned} E(G) &= \sum_{g=1}^8 g \cdot P(G = g) \\ &\approx (1)(0) + (2)(0.00173) + (3)(0.135) + (4)(0.584) + (5)(0.243) + (6)(0.0311) + (7)(0.00475) + (8)(0.000432) \\ &\approx 4.18 \end{aligned}$$

## Evaluation

### Results

In comparison to a strategy entirely focused on solving (Strategy 1), Strategy 2 was entirely focused on gaining information and reducing the size of the consideration list as much as possible. In doing so,

$P(\text{win})$  was increased to 0.995 and  $E(G)$  was lowered to 4.18. The distribution is more tightly clustered about the mean (for instance,  $P(3 \leq G \leq 5) \approx 0.962$ ).

### Improvements to Strategy

While Strategy 1 only guessed words from the consideration list, Strategy 2 essentially only guessed words from outside of the consideration list in order to narrow it down. I soon realized that the issue with this is that in many cases, near the end of the game, it is far better to attempt to solve the puzzle—that is, to make a guess from the consideration list. In my next strategy, I want to find a way to strike a balance between gaining information and guessing from the consideration list, in order to minimize the number of guesses needed to win.

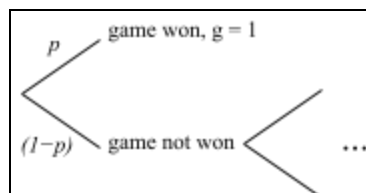
## V. Strategy 3: Expected Score

Because I wanted to see how optimal I could possibly make my algorithm for Strategy 3, I decided to initialize the consideration list to the answer list only (2315 words), instead of the usual guess list (12972 words). Although this reduces the universality of my strategy, I was really curious to see how well my algorithm would perform in optimal conditions.

### *Explanation*

In Strategy 3, I want to combine the guessing and random aspect of Strategy 1 with the entropy component of Strategy 2 into an optimal Wordle algorithm. To do this, I introduced a new variable: **expected score**,  $E(S)$ , where score is synonymous with the number of guesses needed to win. At a given point in a game, the  $E(S)$  of a guess represents the expected total number of guesses needed to win the game if that word is selected as the next guess.

For every guess, there is a probability  $p$  that the guess will be correct and win the game, and a probability  $1 - p$  that the guess will be incorrect (see **Figure 4** below). Even if a guess is incorrect, it is still useful as it will eliminate a portion of the remaining words in the consideration list.



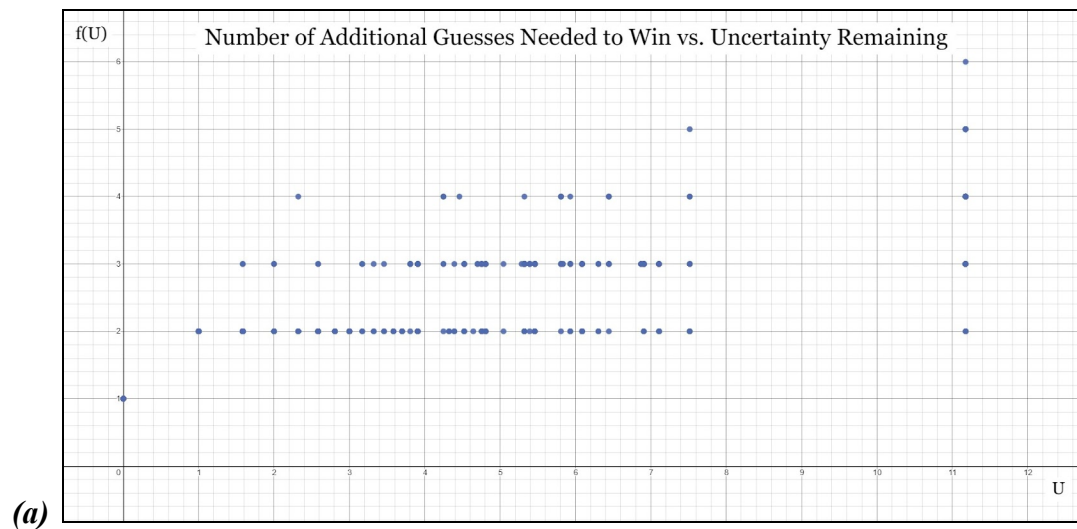
**Figure 4:** Tree diagram outlining the probabilities within a sequence of guesses in Wordle (... to symbolize continuity until the game is won).

**Uncertainty**,  $U$  (in bits), can be understood as the opposite of having information: if there are  $n$  words in the consideration list, then there exists an uncertainty of  $\log_2 n$ . If there were  $U'$  bits of uncertainty before taking a guess that has entropy, or expected information,  $H$ , then the expected amount of uncertainty remaining after taking this guess is given by  $U = U' - H$ , or  $U = \log_2 n - H$ .

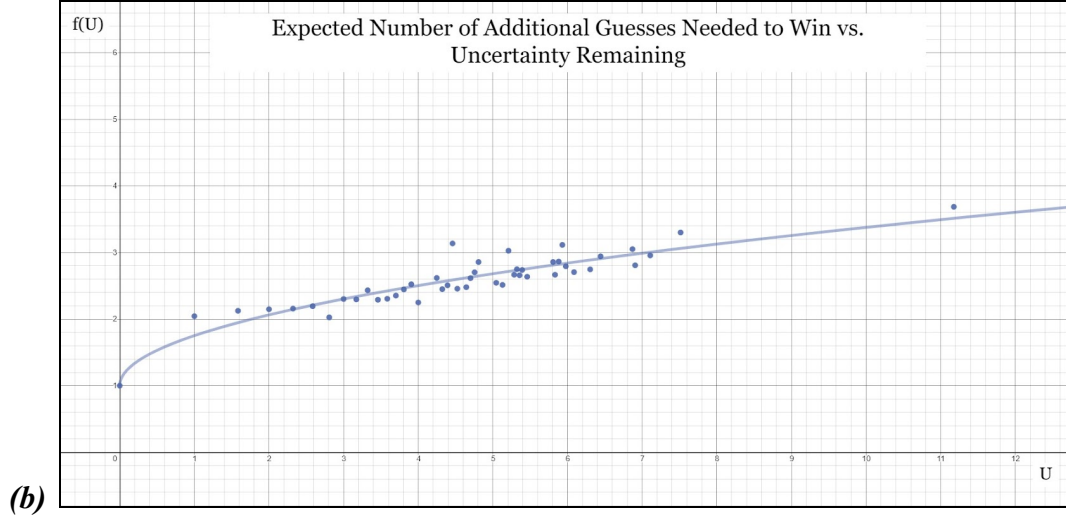
The diagram shown in **Figure 4** can be used to find the expected score of a guess. On the  $g^{\text{th}}$  guess, if the game is won, then the score is  $g$ . Conversely, if the game is not won on the  $g^{\text{th}}$  try, then the number of guesses that still need to be made can be represented as some function of the remaining uncertainty,  $f(U)$ , which will be explored shortly. Thus, the expected score can be expressed as:

$$E(S) = p \cdot g + (1 - p) \cdot (g + f(U)) \quad (\text{Eq. 6})$$

The function  $f(U)$  will be defined to predict how many guesses still need to be made before winning, given how much uncertainty remains in the consideration list. I gathered data to create this function by running Strategy 2 again (modified so that the consideration list is initialized as the answer list) and recording how much uncertainty existed at a certain point in a game and how many more guesses were needed to guess the answer in the end. Plotting these data points, I obtained the graph shown in **Figure 5(a)**. I then vertically averaged out the  $f(U)$  values for each defined value of  $U$  to produce a function from which I could create an equation (**Figure 5(b)**).







**Figure 5: (a) Raw data. (b) Averaged data from (a).**

To find the equation of the curve in **Figure 5(a)**, I first observed that this function very closely resembles a radical function that can be expressed in the form  $f(U) = a\sqrt{U} + k$ . The curve must pass through the point  $(0, 1)$ , since it is certain that only 1 more guess is needed whenever there are 0 bits of uncertainty (i.e.  $2^0 = 1$  option) remaining. Since the parent function,  $f(U) = \sqrt{U}$  has vertical intercept  $(0, 0)$ ,  $k = 1$ . Using trial and error, I found that  $a \approx 0.752$ . Thus,  $f(U) = 0.752\sqrt{U} + 1$ .

To summarize, with Strategy 3, each guess will either attempt to win or gather more information, as the algorithm will decide on the optimal route based on which guess yields the lowest expected score:

$$E(S) = p \cdot g + (1 - p) \cdot (g + f(U)) \quad , \quad \text{where } f(U) = 0.752\sqrt{U} + 1 \quad (\text{Eq. 7})$$

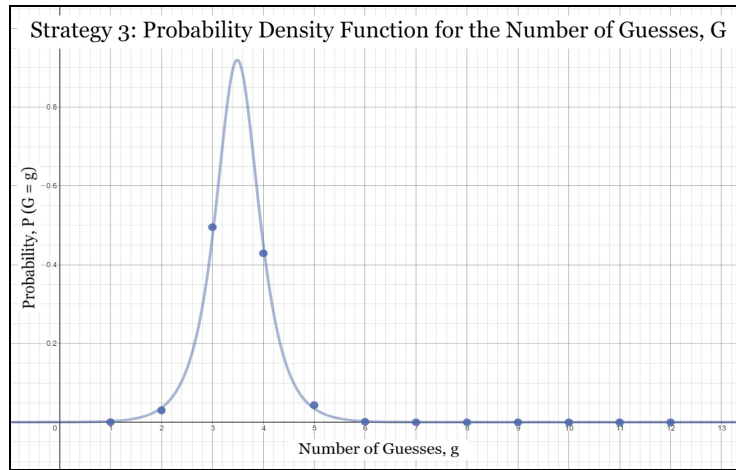
Finally, I realized that with Strategy 3, it was likely that several words may tie for the lowest  $E(S)$ . As in Strategy 1, I decided to use random number generation to randomly select a guess among equally optimal guesses in such cases. I coded this new strategy and found that the best first-guess word according to this algorithm is **RAISE**. A sample run of Strategy 3 is shown in **Figure C1** in **Appendix C**.

### **Data Collection & Processing**

Strategy 3 involves an element of randomness, so I decided to repeat the experiment 5 times and average the results. The full table of raw data is in **Table C1** in **Appendix C**.

**Table 4:** The probability distribution of  $G$ , the number of guesses needed to win Wordle, across 5 trials of 2315 games, for Strategy 3. Data was processed similarly to Strategy 1 (see *Sample calculations 1* and *2*).

$g$	1	2	3	4	5	6
$P(G = g)$	0.000432	0.0302	0.495	0.429	0.0435	0.00130
$g$	7	8	9	10	11	12
$P(G = g)$	0	0	0	0	0	0



**Figure 6:** Probability density function for the number of guesses using Strategy 3. The distribution has been curve-fitted with a transformed hyperbolic secant function:  $f(g) = \frac{1.74}{e^{2.72(x-3.49)} + e^{-2.72(x-3.49)}}$ .

Following *Sample calculations 3* to *6*, I found that:

- $f'(g) = 0$  when  $g \approx 3.49$ . Thus, the maximum is at  $(3.49, 0.870)$ , so  $0 \leq f(g) \leq 1$  for all  $g$ .
- $\lim_{m \rightarrow \infty} \int_{-m}^m \frac{1.74}{e^{2.72(x-3.49)} + e^{-2.72(x-3.49)}} dg \approx 1.00$
- $\lim_{m \rightarrow \infty} \int_{-m}^{6.5} \frac{1.74}{e^{2.72(x-3.49)} + e^{-2.72(x-3.49)}} dg \approx 1.00$

Thus,  $f(g)$  is a valid probability density function,  $P(\text{win}) \approx 1.00$ , and  $E(G) = 3.49$ .

## Evaluation

### Results

Strategy 3 has a 100% win rate and an expected number of guesses of 3.49—I have successfully met my aim of developing a Wordle strategy that performs better than I do! The efficacy of this algorithm can be explained by the fact that it is able to accurately foresee whether it is more optimal to guess directly from the consideration list or to select a high-entropy word to gain more information.

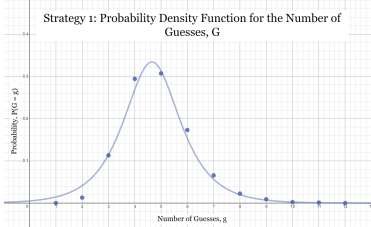
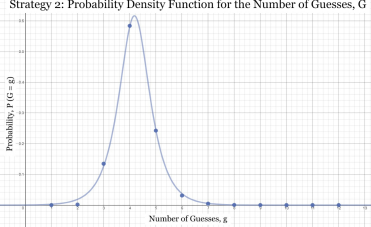
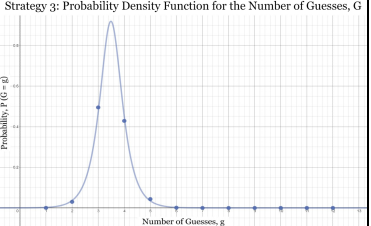
## Methodological Strengths

The  $f(U)$  function I created (Eq. 7) using **Figure 5** was effective at estimating how many additional guesses are required given the amount of uncertainty remaining, which helped the algorithm determine the best course of action. Using random number generation to select a guess among several equally optimal guesses did not lead to significant variation in the results of the 5 trials, but was nonetheless a useful tool for selecting a guess in these situations. With Strategy 3, the proposed continuous probability density function fits the data points very closely and is consequently likely to yield accurate results.

## VI. Conclusion

### Results

**Table 5:** Summary of the results from Strategy 1, Strategy 2, and Strategy 3.

	Strategy 1: Random	Strategy 2: Entropy	Strategy 3: Expected Score
$P(\text{win})$	0.911	0.995	1.00
$E(G)$	4.65	4.18	3.49
<i>Prob. Density Func.</i>			
$f(g)$	$\frac{0.67}{e^{1.05(g-4.65)} + e^{-1.05(g-4.65)}}$	$\frac{1.23}{e^{1.92(x-4.18)} + e^{-1.92(x-4.18)}}$	$\frac{1.74}{e^{2.72(x-3.49)} + e^{-2.72(x-3.49)}}$

Using the two metrics outlined in the **Introduction**, Strategy 3 can be certified as the best of the three, as:

- $P(\text{win}) : 0.911 < 0.995 < 1.00$
- $E(G) : 4.65 > 4.18 > 3.49$

Additionally, Strategy 3 is the most consistent: relative to the parent hyperbolic secant function, Strategy 1 has a horizontal compression factor of 1.05, Strategy 2 has a horizontal compression factor of 1.92, and Strategy 3 has a horizontal compression factor of 2.72, indicating that the probability density function of Strategy 3 is more tightly clustered about its mean (3.49). Greater consistency also relates to higher win rate, as most games span a similar range where the Wordle is solved within 6 guesses.

Through this exploration, I have accomplished my aim of developing and analyzing several Wordle strategies and hence finding an optimal strategy that performs better than I do—my final strategy matched my 100% win rate and surpassed my personal average number of guesses of 3.76.

### ***Methodological Strengths & Limitations***

Using computer programs to simulate and experiment with my different Wordle Strategies was central to the success of this exploration. The programs I created made it possible for me to comprehensively and accurately perform billions of operations on large sets of data, which assures the reliability of my results. Another strength in my methodology was the use of continuous probability density functions to model strategies involving an element of randomness. As the distribution of the values of the random variable  $G$  are not definite in these cases, it is preferable to use a more generalized model, as I have done, to approximate the results of conducting infinite trials. Using these models to subsequently compute  $P(win)$  and  $E(G)$  is hence more reliable than directly using the data collected from the five trials. I believe my models were very good approximations because they fit the criteria for probability density functions: the area under the curve is 1.00 and the range of the function is between 0 and 1, inclusive.

Nevertheless, the models I used are not perfect—the data points do not all fit their respective curves perfectly. I modeled each distribution as a transformed hyperbolic secant function ( $\frac{2}{e^x + e^{-x}}$ ), as it fit my data fairly well and was suitable for integration and differentiation, but this was merely an assumption.

### ***Extensions***

An extension to the current investigation would be to empirically determine the best starting word with Strategy 3 by repeating the experiment with all 12972 possible first guesses and comparing their win rates and mean numbers of guesses. I had chosen RAISE as my starting word when I conducted my experiment because it ranked first when I generated the list of the smallest expected scores for the first guess, but it is likely that the true, empirical results (i.e. when all 2315 games are actually tested) deviate slightly from the expected score values I have calculated. I was unable to run these tests during this exploration due to the extensive amount of time that would be needed—each pass through all 2315 games takes roughly 15

minutes, which amounts to a long time for five trials for each of the 12972 guesses. To shorten the required time, it may be more feasible to only test out the predicted top 100 first guess words, rather than the entire guess list.

Another interesting extension would be to consider relative word frequencies among the guess options. If I had not shrunk my initial consideration list to only contain words from the answer list in Strategy 3, an alternative method could be to assign each word another attribute to each that represents the likelihood that it is the answer, using a binary function (such as a sigmoid function) to distinguish between words that are common/likely to be the answer, somewhat common/somewhat likely to be the answer, and uncommon/unlikely to be the answer.

## ***Takeaways***

Through this exploration, I had the chance to apply knowledge from topics in my AA HL Mathematics course, including Probability & Statistics, Calculus, and Number & Algebra, in a different context. I also learned about new mathematical concepts beyond the scope of the course, such as entropy and information theory. Outside of Wordle, entropy also plays a key role in Machine Learning—the study of algorithms that improve automatically through data acquisition and experience—as minimizing uncertainty and maximizing information is a fundamental goal in the field (Sethneha, 2020). Overall, the use of mathematics helped me accomplish my aim of developing an optimal Wordle strategy and reinforced the validity of my intuitive approach to the game as a human player.

Will this exploration change how I play Wordle from now on? Without the superhuman capacity of knowing the complete answer list and being able to accurately and rapidly filter the consideration list based on obtained feedback, I cannot liken myself to a computer playing Wordle. However, my exploration has helped me better understand the mathematical justification behind the balance between gathering information and guessing the answer. Ultimately, Wordle has served as an excellent gateway into the field of information theory for me and has prompted me to wonder where else I can apply mathematics to facilitate my success in other domains of my life.

## References

- Brownlee, J. (2020, July 13). *A Gentle Introduction to Information Entropy*. Machine Learning Mastery.  
Retrieved from <https://machinelearningmastery.com/what-is-information-entropy/>
- Chow, C. (2022, February 10). *Loaded Words in Wordle*. Medium. Retrieved from  
<https://towardsdatascience.com/loaded-words-in-wordle-e78cb36f1e3c>
- Freshman, C. (2022, March). *Cfreshman's Gists*. GitHub. Retrieved from  
<https://gist.github.com/cfreshman>
- Glen, S. (2021, December 18). *Hyperbolic Secant Distribution*. Statistics How To. Retrieved from  
<https://www.statisticshowto.com/hyperbolic-secant-distribution/>
- Martignon, L. (2001). *Information Theory*. ScienceDirect Topics. Retrieved from  
<https://www.sciencedirect.com/topics/neuroscience/information-theory>
- Sanderson, G. (2022, February 6). *Solving Wordle using information theory*. YouTube. Retrieved from  
<https://www.youtube.com/watch?v=v68zYyaEmEA>
- Sethneha. (2020, November 9). *Entropy: Entropy in machine learning for beginners*. Analytics Vidhya.  
Retrieved from  
<https://www.analyticsvidhya.com/blog/2020/11/entropy-a-key-concept-for-all-data-science-beginners/>
- Wolfram Research. (n.d.). *Introduction to the Hyperbolic Secant Function*. Wolfram Functions. Retrieved  
from <https://functions.wolfram.com/ElementaryFunctions/Sech/introductions/Sech/ShowAll.html>

## Appendix A: Strategy 1

```

SLOPE (answer)

(1) HIMBO
00001
Options remaining: 1946

(2) LOADS
11001
Options remaining: 20
EUSOL GLOST OUSEL SEGOL SKOOL SLOOT SLOPY SLOVE SNOOL SPLOG STOLN SWOLE SWOLN CLOSE SCOWL SLOOP SLOPE SPOOL STOLE STOOL

(3) SWOLN
20210
Options remaining: 8
SKOOL SLOOT SLOPY SLOVE SLOOP SLOPE SPOOL STOOL

(4) SLOOT
22200
Options remaining: 3
SLOPY SLOVE SLOPE

(5) SLOPY
22220
Options remaining: 1
SLOPE

(6) SLOPE
22222
6 tries

```

**Figure A1:** Sample gameplay for guessing SLOPE using Strategy 1 (the consideration list after the first guess was not outputted due to its large size). After guessing (4) SLOOT and receiving feedback 22200, for example, the algorithm updates the consideration list to only include words beginning with “SLO” and not containing ‘T’ or additional instances of ‘O’.

**Table A1:** Raw data for the 5 trials of Strategy 1.

Number of Guesses, $g$	Frequency, $f_g$				
	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
1	0	0	0	0	0
2	46	21	25	20	37
3	309	253	225	231	285
4	685	670	678	670	696
5	675	704	734	755	686
6	364	430	400	398	414
7	149	171	165	157	117
8	55	47	58	54	45
9	23	15	22	19	25
10	6	1	5	7	8
11	3	3	3	4	1
12	0	0	0	0	1

## Appendix B: Strategy 2

```

Result: 00000; Frequency: 858 --> I: 3.91828; p: 0.0661425
Result: 00001; Frequency: 660 --> I: 4.29679; p: 0.0508788
Result: 00002; Frequency: 398 --> I: 5.02649; p: 0.0306815
Result: 00010; Frequency: 288 --> I: 5.49319; p: 0.0222017
Result: 00011; Frequency: 443 --> I: 4.87195; p: 0.0341505
Result: 00012; Frequency: 160 --> I: 6.34119; p: 0.0123343
Result: 00020; Frequency: 82 --> I: 7.30556; p: 0.00632131
Result: 00021; Frequency: 39 --> I: 8.37771; p: 0.00300648
Result: 00022; Frequency: 35 --> I: 8.53383; p: 0.00269812

```

**Figure B1:** The quantity of information  $I$  and the probability  $p$  (calculated by dividing the frequency by 12972, the size of the guess list) for some of the results with a first guess of STARE.

**Table B1:** Some of the results from guessing STARE, sorted in decreasing order. Note that when  $p = 0$ , the calculation  $I = -\log_2(p)$  results in a mathematical error. Hence, if  $p = 0$ , no information can be obtained.

Result	Probability, $p$	Information, $I$ (bits)
00000	0.0661	3.92
00100	0.0638	3.97
10000	0.0626	4.00
...		
01122	0.000154	12.7
02000	0.0000771	13.7
01222	0	
...		

**Table B2:** Comparison between the entropy for a perfectly uniform distribution (same probability across all  $3^5 = 243$  possibilities) and an example of a non-uniform distribution (3 highly likely possibilities and 240 impossible possibilities where  $p = 0$ ).

Uniform Distribution Entropy (bits)	Non-Uniform Distribution Entropy (bits)
$H = -\sum_{i=1}^{243} p \log_2(p)$ $= -\sum_{i=1}^{243} \frac{1}{243} \log_2\left(\frac{1}{243}\right)$ $= -243 \cdot \left(\frac{1}{243}\right) \log_2\left(\frac{1}{243}\right)$ $\approx 7.92 \text{ bits}$	$H = -\sum_{i=1}^3 p \log_2(p)$ $= -\sum_{i=1}^3 \frac{1}{3} \log_2\left(\frac{1}{3}\right)$ $= -3 \cdot \left(\frac{1}{3}\right) \log_2\left(\frac{1}{3}\right)$ $\approx 1.58 \text{ bits}$



**Table B3:** Some of the entropy values for the 12972 possible guesses, sorted in decreasing order.

<b>Guess</b>	<b>Entropy, <math>H</math> (bits)</b>
TARES	6.19
LARES	6.15
RALES	6.11
RATES	6.10
TERAS	6.08
NARES	6.07
SOARE	6.06
TALES	6.05
REAIS	6.05
TEARS	6.03
...	
COCCO	2.38
FUZZY	2.36
FLUFF	2.35
GYPPY	2.31
HYPHY	2.30
PZAZZ	2.29
FUFFY	2.18
IMMIX	2.18
XYLYL	2.13
QAJAQ	2.07

**Table B4:** Raw data for Strategy 2.

Number of Guesses, $g$	Frequency, $f_g$
1	0
2	4
3	312
4	1353
5	562
6	72
7	11
8	1

```

(1) 1 TARES
Options remaining: 814

(2) 2 LOUND
Options remaining: 9
BOGUS COMUS HOCUS MOMUS MOPUS SOJUS WOJUS ZOBUS FOCUS

(3) 3 COMBE
Options remaining: 2
HOCUS FOCUS

(4) 4 AAHED
Options remaining: 1
FOCUS

(5) 5 FOCUS
5 tries

```

**Figure B2:** Sample run of finding the answer FOCUS using Strategy 2.

## Appendix C: Strategy 3

```

BAWDY (answer)

(1) RAISE
Options remaining: 91
BACON BADLY BAGGY BALMY BANAL BANJO BATCH BATON BATTY BAWDY BAYOU CABAL CABBY CACAO CADDY C
A GAMUT GAUDY GAUNT GAWKY GAYLY HANDY HAPPY HATCH HAUNT HAVOC JAUNT JAZZY KAPPA KAYAK LANKY
GAN PAPAL PATCH PATTY TABBY TABOO TACKY TAFFY TALLY TALON TANGO TANGY TATTY TAUNT TAWNY VAU

(2) CYTON
Options remaining: 15
BADLY BAGGY BALMY BAWDY DADDY DALLY GAUDY GAWKY GAYLY HAPPY JAZZY KAYAK MADLY MAMMY PADDY

(3) BADLY
Options remaining: 1
BAWDY

(4) BAWDY
4 tries

```

**Figure C1:** Sample run of finding the answer BAWDY using Strategy 3. Notice that guess 2 seeks to gather information (CYTON is not part of the consideration list, as it does not fit the 02000 feedback from guess 1). Conversely, at guess 3, the algorithm guesses a word from the consideration list (BADLY) instead of waiting to gain more information, as it has deemed this course of action to yield a lower expected score.

**Table C1:** Raw data for the 5 trials of Strategy 3.

Number of Guesses, $g$	Frequency, $f_g$				
	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
1	1	1	1	1	1
2	70	70	70	70	70
3	1148	1148	1144	1147	1148
4	992	993	995	994	992
5	101	100	102	100	101
6	3	3	3	3	3