Jamie Pakutka
Md Amiruzzaman
CSC 576 Data Science
7 May 2022

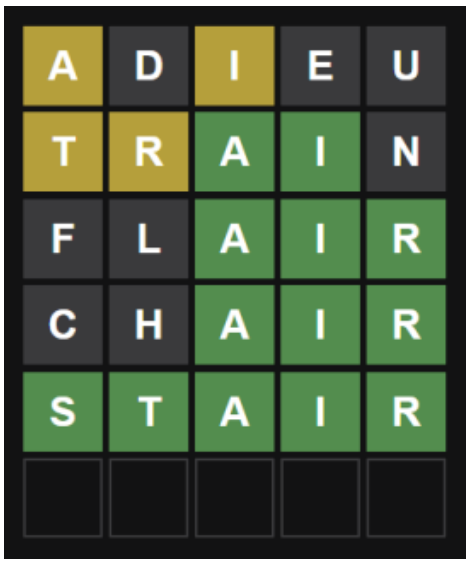
Abstract

For this assignment, I utilized basic list operations and logic to create a simplified solution to solving the daily Wordle. The basis for this solution was from the idea of decision trees, which use a tree-like model to model decisions and their possible consequences. I also included an analysis of what might be the optimal first guess to use.

Introduction:

I viewed several sample data science projects on the internet to try to come up with a concept for this assignment, such as picking out color from images or choosing a tone for a movie scene after scanning the script. With this information, I tried to pick a topic that was more applicable to my life— what about something that I do almost every day?

Wordle is a game published by the New York Times in which the user is given 6 chances to guess a 5 letter word. After inputting an answer, it then responds gray if the letter is incorrect, green if it is correct, and yellow if the letter is present but not in the correct position.



Goal/Problem Statement:

There are a lot of different strategies one can use while solving the daily Wordle, but none of them guarantee a correct answer every time. In this report, I aim to write a program to effectively solve the daily Wordle utilizing Python and its list operations. Afterwards, I hope to use the data found to find the best possible first guess.

Goal:

For each possible correct answer, there is a pattern in which the computer will attempt to guess that word. I aim to write a program to guess the word in a minimum number of

guesses utilizing the dictionary of words I found on Kaggle.

Resources:

In writing program:

Spyder (Anaconda3), Python ver 3.8.8

Wordle

Dataset of Valid Words

In trying to find the best guess:

Word Unscrambler

For article review:

What I Learned from Playing More than a Million Games of Wordle

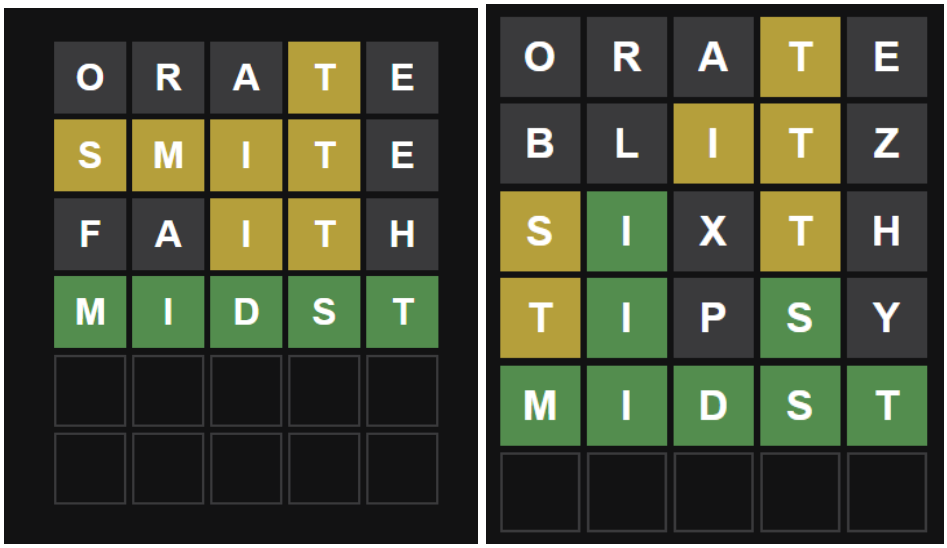
How to Guess Well in Wordle

2. lit review

For the sake of writing my program, I tried to avoid other content on this topic so that I would be unbiased in my product. However, after I finished my program, I read an article from *towardsdatascience.com* by Barry Smyth where he discusses his findings after playing over an excessive amount of games of Wordle. He wrote two articles, the first being "What I Learned from Playing More than a Million Games of Wordle", and the second being "How to Guess Well in Wordle." The first one is an 18 minute read and the other one is 15 minutes, so if you have the time, I greatly recommend checking them out for yourself.

Smyth discusses the basics of Wordle in the first article, such as a statistic of 95% of target words being able to be found in an average of 4 guesses. I do believe this checks out in regards to the results of my program, as you will see below. Smyth's solution differs slightly from mine, as he sums the frequency of each letter to create a score for each word to optimize each guess. He then discusses the significance of your first guess, which he refers to as a "seed" word. Since your first guess will eliminate a lot of future options, it's best to choose a seed word with all unique letters which are all common (words such as "mamma" or "valve" are not optimal picks).

After reading Smyth's article, I found a website that utilizes his method.



My program on the left, word.tips on the right
(wordle on 5/7/22)

On *word.tips*, they gave each word a score based on its probability of being correct and then suggested a list of words to you sorted by likelihood. To see what would happen, I chose the same starting word and chose each word with the highest score as it suggested. It took an extra guess in comparison to my program (explained further below).

I asked some of my friends what they thought of the website in comparison to the program that I wrote, and several of them complained because it doesn't outright give you the best guess, it gives you a list of guesses. This is something that I think gives the website leeway, as they can excuse not getting the right answer as "it was in the list, you just didn't choose it." Therefore in my program I tried to cut out the middleman and take full responsibility for bad suggestions.

3. data

From Kaggle, I downloaded two datasets from the following link: Wordle Valid Words.

Valid Guesses contains: 10658 words.

Valid Solutions contains: 2316 words.

I checked these two files for mutual words, and I was surprised to see that these lists actually had no common words (if you don't count the Header).

For the sake of simplicity, I utilized the smaller dataset (Valid Solutions) to test my code. In this dataset, there are 2316 words (counting the header "Word"). I used python to append all the words into sublists based on which word contains each letter of the alphabet.

Letter	Number of Words containing
a	909
b	267
c	448

d	370
e	1056
f	207
g	300
h	379
i	647
j	27
k	202
l	648
m	298
n	550
o	673
p	346
q	29
r	837
s	618
t	667
u	457
v	149
w	194
x	37
y	417
z	35

4. Analysis & Discussion of Content

I wrote my main program in the mindset of starting with a long list and slowly shortening it until it's almost impossible to get the wrong answer. I tried to picture it like a parse tree, where each decision (guess) that I make leads to smaller and smaller outcomes until I arrive at my destination.

In Wordle, we have 3 subarrays for the letters: Green (right letter, right position), Yellow (right letter, wrong position), and Gray (wrong letter, wrong position).

For each guess, I place the char into their corresponding array. Based on their position, I prune the list accordingly: words containing grey letters are removed whereas words that do not contain green or yellow letters are removed.

I started by hardcoding a correct answer and writing the methods to be able to prune the list of possible solutions. (Well, actually I started by trying to match the valid_solutions dataset to the valid_guess dataset before realizing they had no common words 2 days later, but I prefer not to think about that).

To test my code, I chose to use the list of valid solutions as my data set.

When the program runs, this is an idea of what it looks like:

```
In [3]: runfile('C:/Users/yixin/Downloads/take3.py', wdir='C:/Users/yixin/Downloads')
2316 possible guesses.
Let's try to guess: visor




You guessed howdy
0 0 0 0 0
Grey: h w d y
Yellow: o
Green:
Remaining Possible Words 2315


Cleaning...
897 remaining




You guessed tempo
0 0 0 0 0
Grey: h w d y t e m p
Yellow: o
Green:
Remaining Possible Words 896


Cleaning...
228 remaining




You guessed joist
0 0 0 0 0
Grey: h w d y t e m p j
Yellow: o i s
Green:
Remaining Possible Words 227


Cleaning...
39 remaining
```

```
You guessed conic
0 0 0 0 0
Grey: h w d y t e m p j c n
Yellow: o i s
Green:
Remaining Possible Words 38


Cleaning...
7 remaining


['bison'] ['floss'] ['igloo'] ['kiosk'] ['salon'] ['vigor'] ['visor']


You guessed salon
0 0 0 o 0
Grey: h w d y t e m p j c n a l
Yellow: o i s
Green: o
Remaining Possible Words 6


Cleaning...
2 remaining


['kiosk'] ['visor']


You guessed kiosk
0 i 0 o 0
Grey: h w d y t e m p j c n a l k
Yellow: o i s
Green: o i
Remaining Possible Words 1


Cleaning...
1 remaining


['visor']
Tries: 6
There is logically only one answer left.
Answer: 000o0
```

I tracked the results of this program 10 times. During these trials, I utilized the random class to pick a word for my guess and solution to keep my results unbiased.

I found (calculated in Excel and Spyder):

Most Popular Letters:

e	1056 words
a	909 words
r	837 words

Least Popular Letters:

j	27 words
q	29
z	35

average tries:	5.9 tries
fastest answer:	4 tries
guessed correct word:	4 times
used elimination to find word:	6 times

guess that removed the most words in 10 trials: “troll”

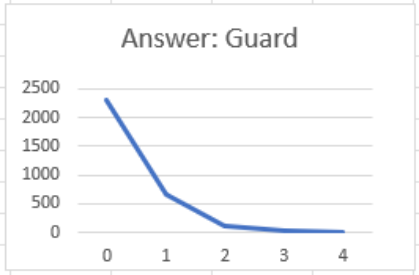
avg words removed after:

1 guess	1596.1 words
2 guesses	2153.4 words
3 guesses	2278.7 words

Here is some of the data followed by a graph that depicts the shrinking list.

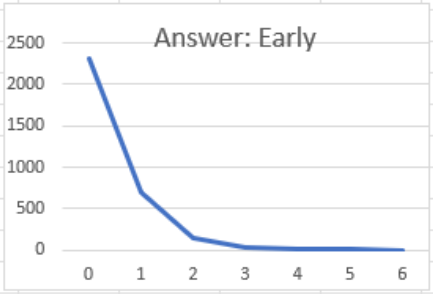
4

Tries	List Size	Guess
0	2316	Alike
1	663	Revue
2	121	Rumba
3	19	Syrup
4	0	Guard



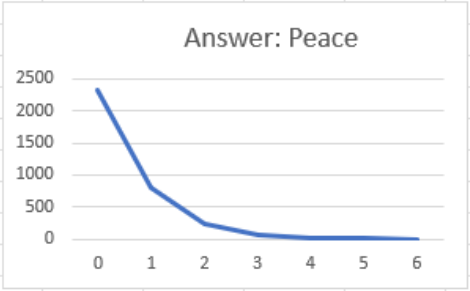
5

Tries	List Size	Guess
0	2316	n/a
1	695	flank
2	151	ovate
3	33	glare
4	14	latte
5	8	maple
6	0	early



6

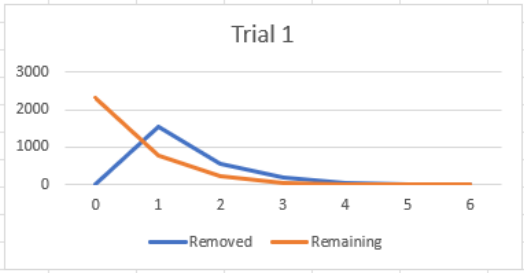
Tries	List Size	Guess
0	2316	n/a
1	790	valve
2	244	steak
3	56	ounce
4	12	cheap
5	3	arena
6	1	craze



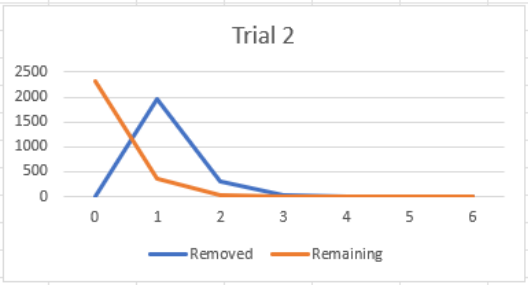
The red font denotes the word being guessed (tables 4 & 5) whereas the bottom table shows an example of the word being found through process of elimination. I noted that “valve” removed less letters than the previous guesses despite containing three popular letters, likely due to the presence of unpopular letter V two times.

Next, I chose a random word as the constant for 5 trials.

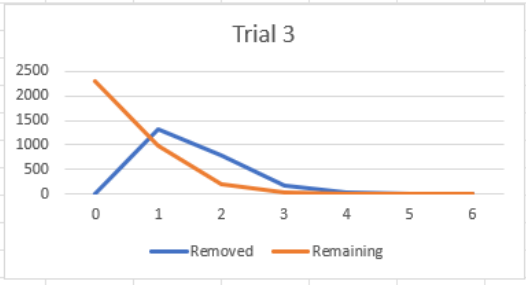
	Removed	Remaining	Word
0	0	2315	n/a
1	1535	780	slain
2	544	236	humph
3	193	43	octet
4	28	15	booby
5	12	3	foggy
6	2	1	goofy



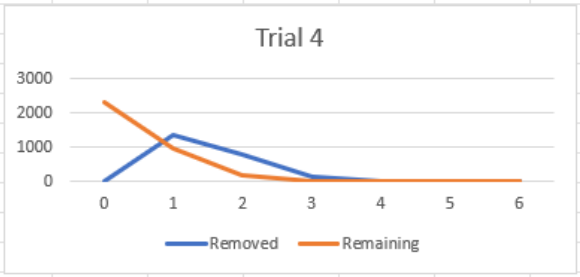
	Removed	Remaining	Word
0	0	2315	n/a
1	1964	351	cagey
2	311	40	rigor
3	29	11	buggy
4	4	7	gypsy
5	4	3	dodgy
6	2	1	goofy



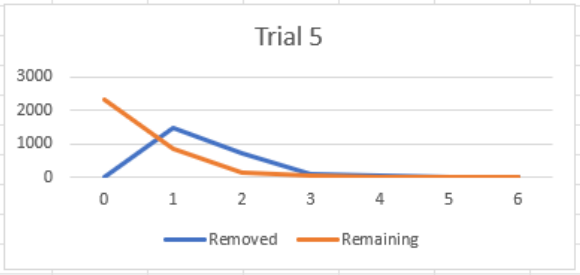
	Removed	Remaining	Word
0	0	2315	n/a
1	1334	981	aphid
2	791	190	eagle
3	160	30	trick
4	26	4	young
5	2	2	mossy
6	1	1	goofy



	Removed	Remaining	Word
0	0	2315	n/a
1	1349	966	etude
2	808	158	elfin
3	140	18	cacao
4	15	3	forgo
5	2	1	foggy
6	1	0	goofy



	Removed	Remaining	Word
0	0	2315	n/a
1	1476	839	spend
2	712	127	relay
3	96	31	amble
4	27	4	quoth
5	3	1	foggy
6	1	0	goofy



Notes:

The guess that removed the most letters was “cagey”, which makes sense because it contains 2 of the previously mentioned Most Popular Letters.

In all 5 trials, it took 6 guesses to find the word. 2/5 tries arrived at the word through a process of elimination whereas the other 3 successfully guessed the word.

Out of 5 trials, “foggy” was guessed 3 times.

Since the previous calculations were done using the randomizer, I tried to see if any of my own guesses would find better First Guesses.

The top 5 most common letters in order of most to least is: e a r o t. So, naturally, I tried to look for words containing these letters to find my best possible answer.

First, I tried to think of my own words, but then I decided to use Old Reliable— Word Unscrambler. E a r o t can be scrambled into orate, oater, and roate (Thanks, Word Unscrambler!) I chose orate (since it was the one that seemed like the most realistic guess someone would think of. Who would guess “oater”?) Next I started adding in letters. The next most common letter was l, so I entered e a r o t l into the Word Unscrambler. This added 9 words to my list: alert, alter, artel, later, ratel, realo, rotal, taler, and tolar.

Note: After reading Barry Smyth’s article, I saw he suggested: tales, cones, trial, hates, round, and climb as best first guesses so I gave each of these a try.

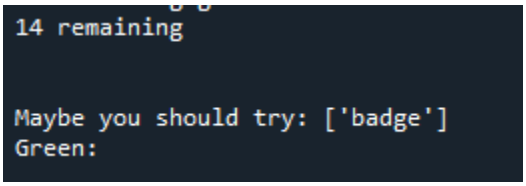
	(Word: avg after 3 tries)		
Orate:	1882	Tales	1810
Alert:	1827	Cones	1820
Rotal:	1771	Trial	1713
Arena:	1669	Hates	1710
Adieu:	1776	Round	1617
		Climb	1694

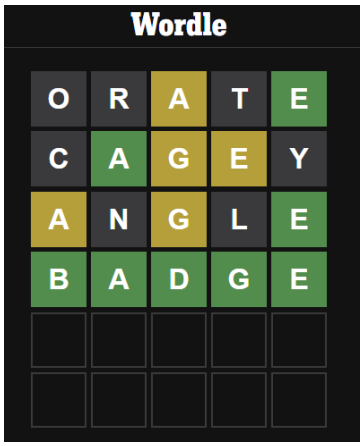
Even after considering Smyth’s article, I still found that orate and alert were the top words for pruning the list.

5. RESULTS

After bug testing for a week, I reformatted my program so that rather than picking a random word and adding the letters to sublists, you can input which letters are green, yellow, or gray so that you can use this program while attempting your own daily Wordle.

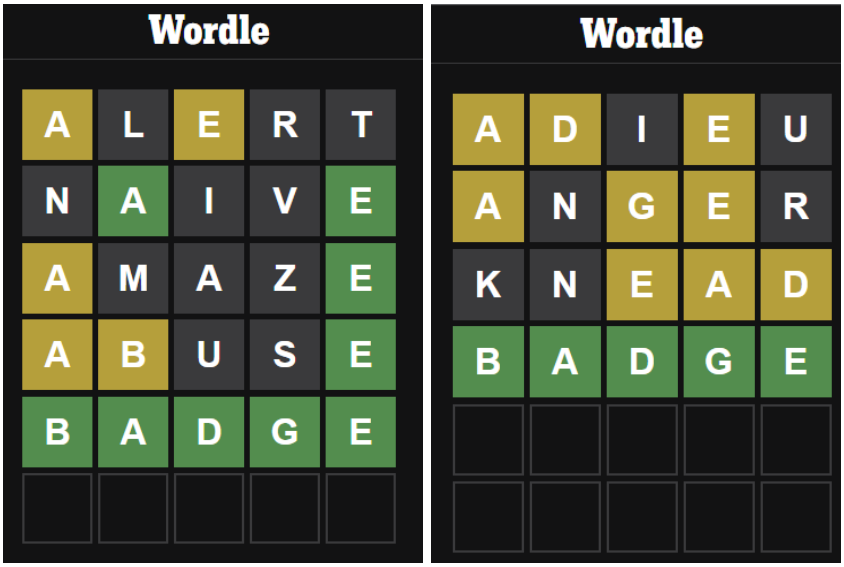
The program then removes all words that contain gray letters along with all words that don’t contain green or yellow letters, just like before! However, now, it suggests a guess at the end from the remaining list.





To test my program, I chose *orate* as my first guess and inputted the results into the console. It suggested a few words, and I solved it on the 4th guess.

I was actually really surprised by the accuracy of the program, despite the fact that I wrote it. Further, I tried again using “alert” since we decided that was one of our best pick from testing. I also used *adieu* because I like that word. It had a better result than “alert” did with this keyword.



(Wordle on 5/6/22)

6. Conclusion

After considering this and reworking my program for 2 weeks, I believe that I did indeed accomplish what I set out to do. I am aware that there are more optimal solutions for this problem. However, the article I reviewed by Barry Smyth confirmed my suspicion that the average word is guessed by attempt 4. My program does satisfy this condition, so I would consider it a win.

In addition to this, I’ve noticed that there is a lot of discussion in the Wordle community regarding the best possible first guess. According to the data I collected, I believe that the best word all around is “orate”, but my personal favorite “adieu” seems to be doing well too.