Optimizing Wordle Guessing: Data Analysis and Strategic Selection

Manit Kaushik
Computer Science - Indraprastha Institute of Information Technology, Delhi

Abstract

This study delves into the realm of word-guessing games, particularly focusing on Wordle, to explore and develop data-driven strategies that enhance the success rates of players. Through meticulous analysis of letter frequencies and index probabilities, a novel scoring system is devised to facilitate optimal initial guesses, thereby increasing the likelihood of early success in the game. Additionally, a strategic seed word selection method is introduced, which effectively reduces the search space while upholding a commendable level of accuracy. The efficacy of these strategies is demonstrated through extensive testing, showcasing their ability to significantly improve the overall success rates of players in solving Wordle puzzles. This research underscores the importance and effectiveness of integrating data-driven insights and strategic thinking in the development of successful strategies for word-guessing games like Wordle.

Introduction

Wordle presents an intriguing challenge where players must decipher a concealed five-letter word within six attempts. With each guess, the game offers feedback by color-coding the letters in the guessed word: green signifies a correct letter in the correct position, yellow indicates a correct letter but in the wrong position, and grey denotes an incorrect letter. The game's essence lies in strategic guessing and elimination, compelling players to blend linguistic intuition with deductive reasoning to unravel the target word within the limited attempts.

The initial guess in Wordle often involves a leap of faith, a blind attempt to crack the code. However, consider the possibility of strategically enhancing our likelihood of receiving yellow or green feedback on the first attempt, or perhaps within the initial 2-3 tries, through mathematical methods

Data Analysis for Guessing Optimization

One effective initial guessing strategy is to select a word that contains the most commonly occurring letters. Each letter’s frequency in this word is determined by the total number of times that letter appears in all the possible words in our chosen dataset. Here, we are dealing with two datasets. One is the valid solution of the Wordle puzzle, which has 2,315 words; the other is all possible five-letter words in the English Dictionary. Essentially, all the valid guesses that the Wordle puzzle can take are approximately 13,000 words. We aim to increase the probability of getting yellow and green letters on our first try. So, it would make sense to use the valid guesses dataset to look for the most common occurrences.

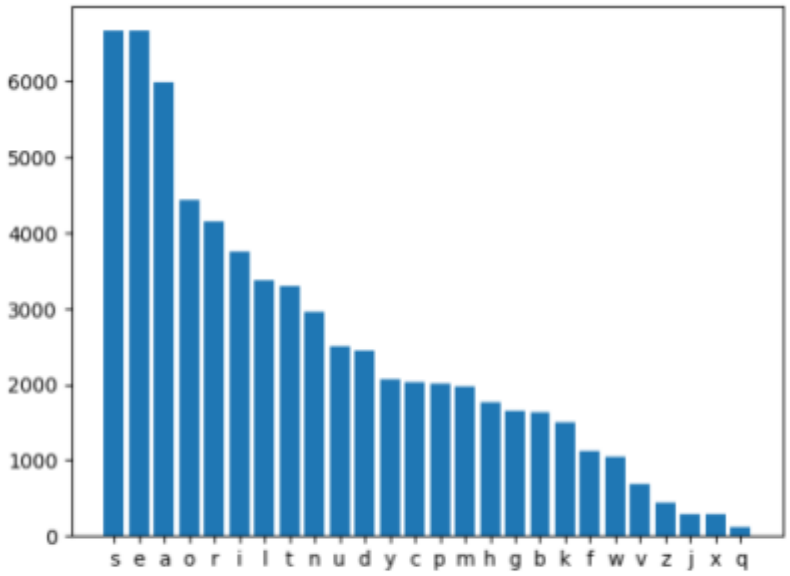

Fig1. Alphabet Frequency Graph

The first step to achieve our goal is to see the **alphabet frequency** for each letter, i.e. computing all the times an alphabet appears in the five-letter words.

By looking at the above graph, we know which letters to use to increase our probability of yellow letters. However, to increase the probability of achieving any green letter, we should also focus on the frequency of a letter at a given index, namely the **letter index frequency.** A letter can only occur at five indexes in a letter word. The probability of a letter occurring in any of these five indexes is calculated by reviewing the dataset of valid guesses and seeing where each letter occurs while incrementing its letter index frequency for that particular index. Here is the heatmap for the letter index frequency and values of letter index probability for all the letters.

```
manitk@ubuntu:~$  python letter_index_prob.py
     Letter Index Probability
    Index: 0    1    2    3    4
a - [0.12, 0.38, 0.21, 0.18, 0.11]
b - [0.56, 0.05, 0.21, 0.15, 0.04]
c - [0.45, 0.09, 0.19, 0.2,  0.06]
d - [0.28, 0.03, 0.16, 0.19, 0.34]
e - [0.05, 0.24, 0.13, 0.35, 0.23]
f - [0.54, 0.02, 0.16, 0.21, 0.07]
g - [0.39, 0.05, 0.22, 0.26, 0.09]
h - [0.28, 0.31, 0.07, 0.13, 0.21]
i - [0.04, 0.37, 0.28, 0.23, 0.07]
j - [0.69, 0.04, 0.16, 0.1,  0.01]
k - [0.25, 0.06, 0.18, 0.33, 0.17]
l - [0.17, 0.21, 0.25, 0.23, 0.14]
m - [0.35, 0.1,  0.26, 0.2,  0.09]
```

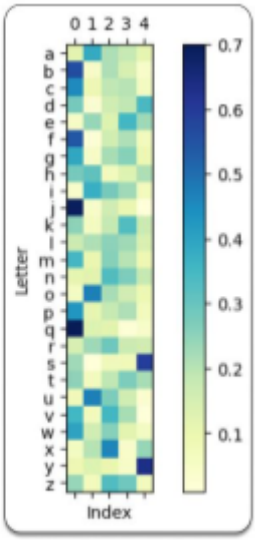

Fig 2. Letter Index Probability Values & Heatmap

Now, to improve the score for each word, we combine the Letter Index Probability (LIP) with the Alphabet Frequency (AF) in such a way that -

$$Score\ of\ a\ Word\ (\alpha_1\alpha_2\alpha_3\alpha_4\alpha_5) = \Sigma\ LIP(\alpha_i) * AF(\alpha_i)$$

We then pivot our dataset for only valid solutions so that our starting word is a solution. We calculate the score for each word in the valid solutions dataset. This is the final list based on the decreasing order of scores.

```
manitk@ubuntu:-$ python sorted_scores.py
 Word   Score
1. basis: 8517.44
2. torus: 8450.3
3. bonus: 8305.72
4. saner: 7780.45
5. boney: 7609.93
6. crass: 7592.62
7. brass: 7591.14
8. focus: 7407.86
9. minus: 7390.7
```

Fig 3. Word Scores

Strategic Seed Word Selection

The above work was based on maximizing **alphabet frequency & letter index probability.** To further improve our chances of solving the Wordle, we will apply another strategy commonly used by the players. This involves using the first three tries to input three fixed **seed words,** which are unique to each other, have no common letters, and cover all the vowels. This strategy helps the players guess the answer to the puzzle in the next tries. This is because, most of the time, players get many critical cues due to the diverse nature of the seed words. One possible seed word set is **Alter, Bison, and Duchy.**

Before using this strategy, we must remove words in the score list with repeating letters. This is necessary since we want the number of unique letters in the three seed words to be as high as possible, increasing the probability of getting yellow letters if not green letters. The above list, after doing this operation, looks like this.

```
manitk@ubuntu:-$ python sort_unq_scr.py
 Word   Score
1. torus: 8450.3
2. bonus: 8305.72
3. saner: 7780.45
4. boney: 7609.93
5. focus: 7407.86
```

Fig 4. Decreasing Word Scores

Moving on, we now create combinations of these words remaining in the scores list, such that the combination of three words has no common letters. This idea follows the main idea of our strategy. Simultaneously, we should also calculate the sum total score for that particular combination of words by adding the scores for each word in that combination. This will help us get the seed words with the highest possible score and are diverse, hence maximizing the probability of getting yellow and green letters in our first 3 tries. These are possible seed words listed in descending order of their total score -

```
manitk@ubuntu:-$ python seed_words_cmb.py
   Seed Words Combination       Score
1. ('bonus', 'camel', 'dirty'): 20292.25
2. ('bonus', 'cadet', 'girly'): 20263.81
3. ('bonus', 'warty', 'plied'): 20194.96
4. ('minus', 'party', 'bowel'): 20008.94
5. ('torus', 'camel', 'pinky'): 19992.93
```

Fig 5. Potential Seed Words

From the above list, we finish with the words **Bonus, Camel, and Dirty** being our seed words, i.e. our first 3 tries. For our Wordle Solver, the program will always use its first 3 tries as **Bonus, Camel, and Dirty.** So far, we have reduced our search space to the minimum possible solutions. But doing so has cost us three tries out of six. And our search space for potential solutions that fit our current list of yellow and green letters may have more than three words. Hence, a simple brute force method is not going to work. Each word left in the search space has a corresponding score assigned in previous steps. We will take the word in the search space with the maximum score for each of the next guesses and use that as our next try. Then, using the new information, such as new greyed-out letters and potentially new yellow and green letters, we eliminate the words in the search space and repeat the process until we reach the answer or the guesses are exhausted. Let's see how our Wordle solver strategy fairs in the game. We test how the program handles every possible solution, from the 2315 solutions to the puzzle.

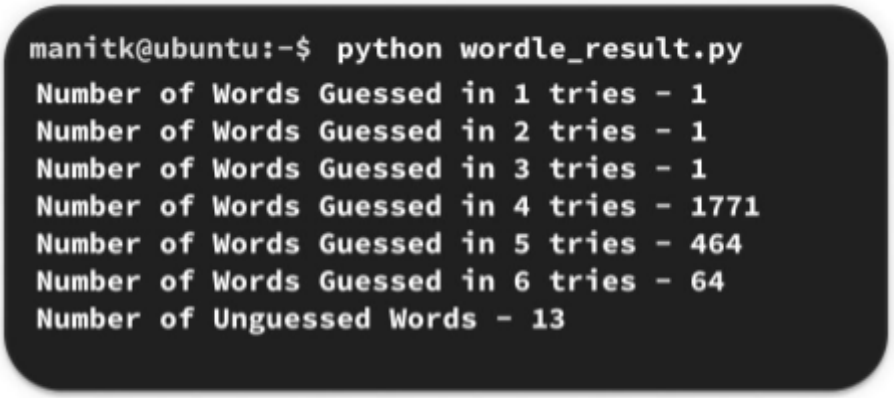

Fig 7. Final Results

Out of all the possible solutions to Wordle, our program solved for every word except for 13. That's an accuracy of 99.43%! The words the program couldn't guess in six tries, or less were **eager, eater, hover, jaunt, joist, jolly, rover, stave, vaunt, voter, wafer, waste, and wound.**