Selecting Optimum Seed Words for Wordle using Character Statistics

Nisansa de Silva
Department of Computer Science & Engineering
University of Moratuwa
Moratuwa, Sri Lanka
NisansaDdS@cse.mrt.ac.lk

2022 Moratuwa Engineering Research Conference (MERCon) | 978-1-6654-8786-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/MERCon55799.2022.9906176

***Abstract*—Wordle, a word guessing game rose to global popularity in the December of 2021. The goal of the game is to guess a five-letter English word within six tries. Each try provides the player with hints by means of colour changing tiles which inform whether or not a given character is part of the solution as well as, in cases where it is part of the solution, whether or not it is in the correct placement. Numerous attempts have been made to find the best starting word and best strategy to solve the daily *Wordle*. This study uses character statistics of five-letter words to determine the best three starting words. We show that our proposed starting words perform better than the currently available suggested word configurations across three distinct data sets: of which, two are drawn from the official *Wordle* word database.**

***Keywords*—Wordle, Character Statistics, Word-game, Optimisation, Puzzle Solving**

I. Introduction

The web-based word game *Wordle* [1], since its introduction in November of 2021 has captured the minds of the Internet. The game-play is rather simple. It gives players six attempts to guess a five letter word. At each attempt, hints are given to the player with coloured tiles. The allure of the game is one part due to the false scarcity it has created by limiting it to one puzzle per day. The analysis on value of the skills developed by engaging in this daily mental activity in fields such as medicine has already started [2]. The New York Times, bought over *Wordle* in January 2022 to capitalise on its millions of daily users [3, 4]. Given the popularity of the game [3–5], there have been a number of attempts to find the best first word for fast and accurate solving of the daily puzzle. The work by Kandabada [6] suggested manually selected four words, `[SPORT, CHEWY, ADMIX, FLUNK]`, as the best starting words. The work by Sidhu [7] attempted to find the best starting word from a linguistic perspective while the work by Horstmeyer [8] suggested to do the same by running over 1 million simulations. Bram and Cardin [9] have proposed two strategies to employ in winning the game which they derived from their experience with crosswords. Interestingly, there also have been studies done on the worst word with which to start the wordle [10]. Anderson and Meyer [5] have used machine learning to find the optimal human strategy for solving the *wordle*. The work by Short [4] proposed Most Rapid Decrease (MRD) algorithm and Greatest Expected Probability (GEP) algorithm to determine the best starting word. Both the algorithms predicted it to be `TARES`.

The objective of this work is to derive the set of 3 optimum starting words for *wordle* covering 15 different characters and ordered in the descending order of significance. The rest of the paper is organised as follows, Section II provides a brief introduction to *Wordle*, Section III describes our methodology, Section IV reports our experiments and results. Finally, Section VI concludes the paper.

II. Background

The game accepts 12972 words as possible guesses for solutions while it has 2315 secret words as the actual solutions [5]. The tile based hints given to the player are as follows:

1) **Green:** The entered character is in the expected solution and is in the expected position.
2) **Yellow:** The entered character is in the expected solution but it is not in the expected position.
3) **Gray:** The entered character is not in the expected solution.

Figure 1 shows four examples of *wordle* solutions with the tile colours providing hints to the player to progress. Note from Fig 1d and Fig 3b that it is possible for the solution to have repeated characters. Also observe from Fig 3b how these repeated characters are not clued in using the colour codes. The first line clues that *E* is in the solution. But there is no indication as to how many times the character would appear. Another point to note is from Fig 1c, where it can be seen that *Wordle* uses American spelling despite being developed by a person of UK origin and hosted at a `.co.uk` web address. Full automatic solving of *Wordle* is a constraint satisfactory problem similar to that of the work by de Silva et al. [11].

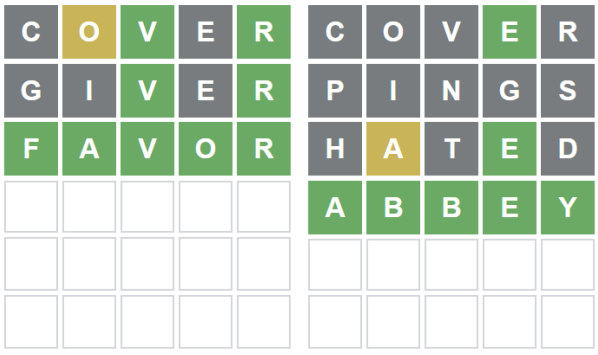
III. Methodology

The following novel methodology is proposed by us. We obtain a manually prepared word list G and derive the *all word list*, A as shown in Equation 1 where, the $len(\cdot)$ function gives the length of the word w. This removes all words which are not of length 5 (e.g., *cat*, *dragons*).

$$A = \bigcup_{w \in G} \begin{cases} \{w\} & \text{if } len(w) = 5 \\ \emptyset & \text{otherwise} \end{cases} \quad (1)$$

978-1-6654-8786-3/22/$31.00 ©2022 IEEE

(a) Guessing the word *Drink* (b) Guessing the word *Tangy*



(c) Guessing the word *Favor* (d) Guessing the word *Abbey*

Fig. 1: Few Examples of *Wordle* Solutions

Next, we initialise the *character frequency map*, F of the format `<character,value>` as shown in Equation 2, where C is the alphabet.

$$\underset{c \in C}{\forall} \Big[F(c) = 0 \Big] \tag{2}$$

Then, we populate the *character frequency map*, F as shown in Equation 3, where A is the all word list, C is the alphabet, and c_2 is a character in the word w.

$$\underset{w \in A}{\forall} \left[\underset{c_1 \in C}{\forall} \left[F(c_1) = F(c_1) + \sum_{c_2 \in w} \begin{cases} 1 & \text{if } c_1 = c_2 \\ 0 & \text{otherwise} \end{cases} \right] \right] \tag{3}$$

Next, we update the *character frequency map*, F, as shown in Equation 4, where A is the *all word list*, C is the alphabet, and the $len(w)$ gives the character count (length) of the word w. This results in F registering the global frequencies of each of the characters in the alphabet.

$$\underset{c_1 \in C}{\forall} \left[F(c_1) = \frac{F(c_1)}{\sum\limits_{w \in A} len(w)} \right] \tag{4}$$

We define the *unique word list*, W as shown in Equation 5, where A is the *all word list*, c is a character in the word w, and the $len(\cdot)$ function gives the size of the set. This function makes sure that the W only contains words that have five unique characters. This removes 5 letter words which have repeated characters (e.g., *feels*).

$$W = \bigcup_{w \in A} \begin{cases} \{w\} & \text{if } len\big(\bigcup\limits_{c \in w} \{c\} \big) = 5 \\ \emptyset & \text{otherwise} \end{cases} \tag{5}$$

We define the *word value map*, M of the format `<word,value>` as shown in Equation 6, where W is the *unique word list* from Equation 5, c is a character in word w and $F(c)$ is the value stored in the *character frequency map* (created in Equation 4) for the character c.

$$\underset{w \in W}{\forall} \left\{ M(w) = \sum_{c \in w} F(c) \right\} \tag{6}$$

Next, we define word overlap as shown in Equation 7 where w_1 and w_2 are the candidate words and i is a character. Thus, I_{w_1,w_2} will carry the Boolean value `TRUE` is there is at least one common character between w_1 and w_2 or carry the Boolean value `FALSE` otherwise.

$$I_{w_1,w_2} = \exists i \text{ s.t } i \in w_1 \wedge i \in w_2 \tag{7}$$

Next we define a greedy algorithm to select the current *best words set* as shown in Equation 8 where, B is the set of best words, B_0 is the first element of B, and M is a *word value map* of the format `<word,value>` (including but not limited to that which was defined in Equation 6). This process returns, as the result of $B(M)$, the highest valued words in M.

$$B(M) = \underset{w \in keys(M)}{\forall} \begin{cases} \{w\} & \text{if } B = \emptyset \text{ or } M(w) > M(B_0) \\ B \cup \{w\} & \text{if } M(w) = M(B_0) \\ B \cup \emptyset & \text{otherwise} \end{cases} \tag{8}$$

We define the *simplified best word list* as shown in Equation 9 where L is a list of words, L_0 is the first word in L, $\{L_1, \dots L_n\}$ is the list of words in L other than L_0, and n is the number of words in L. What this does is, given an L, it removes each of the words in $\{L_1, \dots L_n\}$ which has character overlaps with L_0 in an iterative manner and keeps the rest

$$S(L) = \begin{cases} L & \text{if } n \leq 1 \\ L_0 \cup S\left(\bigcup\limits_{l \in \{L_1, \dots L_n\}} \begin{cases} \emptyset & \text{if } I_{L_0,l} \\ l & \text{otherwise} \end{cases} \right) & \text{otherwise} \end{cases} \tag{9}$$

We define the *filtered word value map*, $M^{'}$ of the format `<word,value>` as shown in Equation 10, where M is a *word value map* of the format `<word,value>` (including but not limited to that which was defined in Equation 6) and w_1 is a given filter word. This algorithm makes sure that $M^{'}_{M,w_1}$ contains the subset of `<word,value>` pairs from M such that, none of the keys have a character overlap with w_1.

$$\underset{w_2 \in keys(M)}{\forall} \left\{ M^{'}_{M,w_1}(w_2) = M(w_2) \text{ if } I_{w_1,w_2} = \text{ FALSE} \right\} \tag{10}$$

Finally we define candidate processing in Equation 11, where where M is a *word value map* of the format

<word,value> (including but not limited to that which was defined in Equation 6), the function $B(\cdot)$ is as defined in Equation 8, the function $S(\cdot)$ is as defined in Equation 9, the *filtered word value map*, $M^{'}$ is as defined in Equation 10, and w is a word.

$$P(M) = S(B(M)) \underset{w \in S(B(M))}{\cup} \left[P(M^{'}_{M,w}) \right] \tag{11}$$

IV. Experiments

For the manually prepared word list G we used a publicly available word list from github[1] which has over $466k$ English words. From the obtained G, using Equation 1, we derived the *All word list*, A. We noted that A contains only 21952 words. Also, for the benefit of Equation 4, we calculated the total number of characters in the words in A and observed that it has 109760 characters. We show in Table I, the character frequencies we calculated for the *character frequency map*, F as shown in Equation 4.

TABLE I: Calculated Character Frequencies

Character	Frequency	Character	Frequency
a	0.1124	n	0.0546
b	0.0268	o	0.0653
c	0.0330	p	0.0263
d	0.0355	q	0.0015
e	0.0994	r	0.0648
f	0.0147	s	0.0754
g	0.0236	t	0.0491
h	0.0290	u	0.0399
i	0.0661	v	0.0114
j	0.0057	w	0.0135
k	0.0224	x	0.0042
l	0.0556	y	0.0314
m	0.0318	z	0.0069

An interesting observation we can make from Table I is that the character a is more frequent than character e. This contrasts the character frequency behavious reported in earlier literature [12, 13]. Other than that, all the other character frequencies aligns with common wisdom. After using Equation 5 to obtain W, we observed to have left with 13672 words. This is 62.28% of the word count we had in A. It is an interesting to observe that in the 5 letter word domain, majority of the words seem to have 5 unique characters rather than having repeated characters. Figure 2 shows a part of the *word value map*, M of the format <word,value> as generated in Equation 6. Note that this even contains proper names (e.g., abdul), which the current version of Wordle does not have as solutions. This is kept as is for the sake of generalisation.

When we executed Equation 11, and extracted our word suggestions. At this point, we faced with a problem. It was the fact that some of the words that were suggested as candidates were not being accepted as valid words by *Wordle*. This, we observe, is due to the fact that our word list is richer than that used in *Wordle*. Because of this, we had to manually drop a number of high ranking words. The words that we had to drop

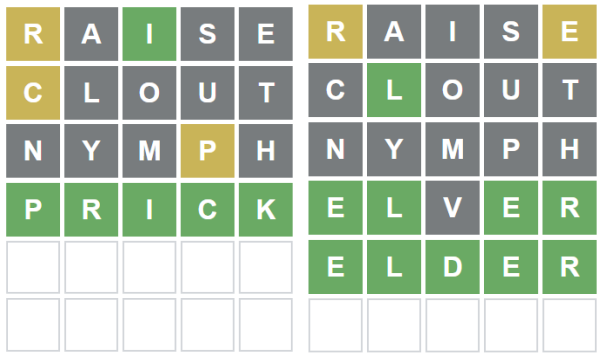
[1] https://github.com/dwyl/english-words

```
{'abdel': 0.329701166180758,
 'abdom': 0.27143768221574344,
 'abdon': 0.29451530612244897,
 'abdul': 0.2702259475218659,
 'abend': 0.32860787172011663,
 'abert': 0.3523870262390671,
 'abets': 0.36307397959183674,
 'abhor': 0.2982325072886297,
 'abide': 0.34013301749271135,
 'abied': 0.3401330174927114,
 'abyed': 0.305493804664723,
 'abies': 0.38010204081632654,
 'abyes': 0.34546282798833816,
 'abihu': 0.27418002915451894,
```

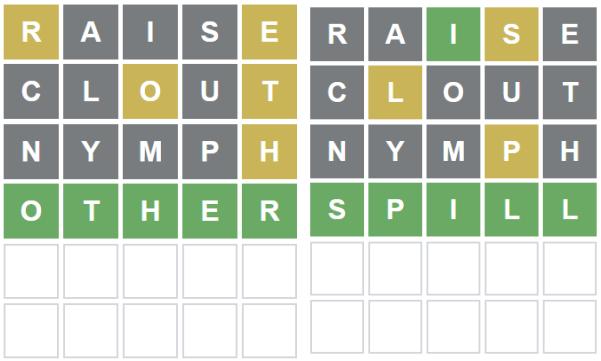
Fig. 2: A part of the *Word Value Map*

are: *aires*, *erisa*, *saire*, *luton*, *tould*, *unold*, *dunlo*, *xdmcp*, and *aries*.

The highest ranking suggestion to be accepted by Wordle was ['serai', 'nould']. However, this had to be abandoned due to having only two words and thus only covering 10 characters. The highest ranked 3 word set accepted by wordle was ['aesir', 'donut', 'lymph']. Logically, these are the best three words to use. However aesir is not a very common word used by our alpha testers. Thus, we opted to settle with the next best word list ['raise', 'clout', 'nymph']. Figure 3 shows two examples of solved *Wordle*s with the selected words.

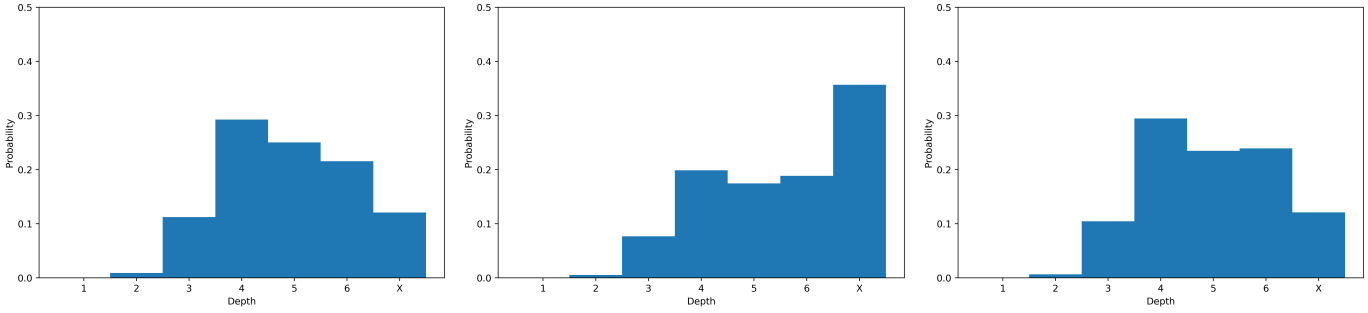


(a) Guessing the word *Prick* (b) Guessing the word *Elder*

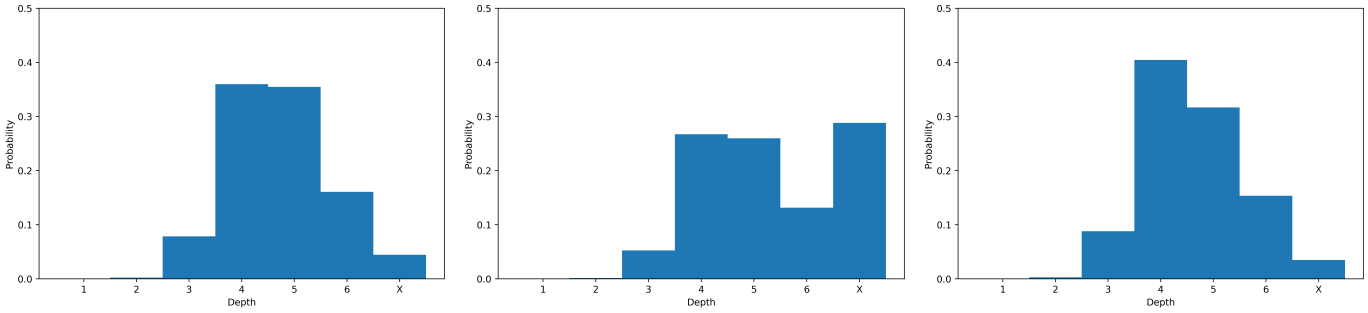


(c) Guessing the word *Other* (d) Guessing the word *Spill*

Fig. 3: Four Examples of *Wordle* Solutions with the selected words ['raise', 'clout', 'nymph']



(a) (No Seed Words,Training Data Set) (b) (No Seed Words,Wordle Source Data Set) (c) (No Seed Words,Wordle Target Data Set)



(d) (Our Seed Words,Training Data Set) (e) (Our Seed Words,Wordle Source Data Set) (f) (Our Seed Words,Wordle Target Data Set)

Fig. 4: Simulation Results of *Wordle* with different data sets as well as with and without our proposed seed words, `['raise', 'clout', 'nymph']`. The depth indicates the depth at which a puzzle is solved and the special depth X indicates instances where a solution was not found within the allotted depth of 6 attempts. Each of the graphs are captioned following the convention (S, D) where, S describes whether no seed words were used or our proposed seed words were used, and D indicates what data set was used to generate the graph.

V. Results

In order to further evaluate the effectiveness of our proposed seed words, we decided to collect data on *Wordle* runs. However, given that the official *Wordle* only releases one puzzle per day, it is inadequate to evaluate our seed words and compare them to those in the literature. Hence it was decided to take the simulation route as proposed by Horstmeyer [8]. In order to build a simulator, we first recreated the *Wordle* solver proposed by Petersen [14]. However, then we noted that, while the implementation removes yellow tagged characters from the relevant position, it does not subsequently enforce the fact that the said character should be included in the solution at a different location. This results in a larger search space for the following iterations and hinders the convergence within the expected maximum depth. In our implementation, we added this enforcement. Figure 5 shows the ablation study.

Then, we replaced the human input component with a responsive model of the *Wordle* feedback system. Finally, we modified the solver algorithm to accept external seed words. Further, from this point onward, the word list we used to generate our seed words in Section IV shall be referred to as the *Training Data Set*. In addition to this data set, we obtained the list of all possible future and past puzzle target words as well as the list of all accepted source words from the official *Wordle* app [1]. These are named *Wordle Target Data Set* and *Wordle Source Data Set*, respectively. As mentioned in Section IV, the training data set contains 21952 words. The new *Wordle Target Data Set* and *Wordle Source Data Set*, respectively contain 2309 and 10638 words. Figure 4 shows the result of the simulations with and without our seed words for all three data sets. From these graphs it is evident that the *Wordle Source Data Set* is the hardest to tackle and that in all scenarios (including the *Wordle Source Data Set*), the model with our seed words have a higher chance of solving the puzzle (i.e., getting a solution within depth 1 to 6). The graphs also show that comparatively, our seed words arrive at the solution at a lower depth than the *no seed* configuration.

Next, we wanted to compare our seed words against the seed words proposed by prior work [6–8, 10, 15–17]. To achieve this end, we ran a total of 1.8 million simulations (surpassing the 1 million simulations conducted by Horstmeyer [8]). We report the result of these experiments in Tables II, III, and IV. In each of the tables we have reported the baseline of *no seed words* as discussed in Fig 4 and proposed by Petersen [14], as well as the results of using one, two, or all three of our proposed seed words. The latter analysis is important given that Neil [17] and Groux [16] propose the same starting seed word, `'raise'`, as we do.

In order to better model how a human will be solving *Wordle*, we have also introduced an exception to the configurations that use seed word sets which contain more than one word. In the cases where the the simulation has used one or more seed

TABLE II: Simulation Results on the *Training Data Set*

Seed Words	Count							Cumulative Percentage					
	1	2	3	4	5	6	X	1	2	3	4	5	6
No seed [14]	1	196	2464	6418	5492	4731	2650	0.00	0.89	12.11	41.35	66.37	87.92
sport chewy admix flunk [6]	1	51	1179	3880	9750	5957	1134	0.00	0.23	5.60	23.27	67.69	94.83
soare [7, 15]	0	190	2433	6496	5452	4668	2713	0.00	0.87	11.95	41.54	66.38	87.64
adept clamp plaid [8]	1	43	498	1574	6560	7869	5407	0.00	0.20	2.47	9.64	39.52	75.37
slice [8]	1	181	2449	6853	5663	4557	2248	0.00	0.82	11.98	43.20	69.00	89.76
tried [8]	1	182	2455	6949	5871	4422	2072	0.00	0.83	12.01	43.67	70.41	90.55
crane [8]	1	182	2477	6696	5654	4448	2494	0.00	0.83	12.11	42.61	68.37	88.63
tares [10]	1	209	2598	6640	5466	4494	2544	0.00	**0.95**	12.78	43.03	67.93	88.40
lares [10]	1	201	2539	6675	5429	4519	2588	0.00	0.92	12.49	42.90	67.63	88.22
rales [10]	1	198	2511	6732	5417	4492	2601	0.00	0.90	12.34	43.01	67.69	88.15
rates [10]	1	202	2549	6609	5516	4510	2565	0.00	0.92	12.53	42.64	67.77	88.31
cares [10]	1	208	2442	6631	5569	4547	2554	0.00	**0.95**	12.07	42.28	67.65	88.36
slate [16]	1	196	2538	6760	5578	4486	2393	0.00	0.89	12.45	43.24	68.65	89.09
reais [16]	0	195	2456	6367	5422	4688	2824	0.00	0.89	12.08	41.08	65.78	87.14
arose [16]	1	178	2418	6391	5600	4649	2715	0.00	0.81	11.82	40.93	66.44	87.62
raise [16, 17]	1	194	2414	6361	5582	4727	2673	0.00	0.88	11.88	40.86	66.29	87.82
raise clout	1	48	2805	7533	5797	4085	1683	0.00	0.22	**13.00**	**47.32**	73.73	92.34
raise clout nymph	1	48	1721	7893	7786	3526	**977**	0.00	0.22	8.06	44.02	**79.49**	**95.55**

TABLE III: Simulation Results on the *Wordle Source Data Set*

Seed Words	Count							Cumulative Percentage					
	1	2	3	4	5	6	X	1	2	3	4	5	6
No seed [14]	0	55	816	2114	1856	2004	3793	0.00	0.52	8.19	28.06	45.51	64.35
sport chewy admix flunk [6]	0	29	428	1449	3557	2180	**2995**	0.00	0.27	4.29	17.91	51.35	**71.84**
soare [7, 15]	1	58	818	2122	1808	1987	3844	0.01	0.56	8.25	28.20	45.20	63.88
adept clamp plaid [8]	0	15	161	646	2225	2814	4777	0.00	0.14	1.65	7.72	28.64	55.09
slice [8]	0	48	757	2247	1939	1984	3663	0.00	0.45	7.57	28.69	46.92	65.57
tried [8]	0	82	861	2277	2000	1881	3537	0.00	0.77	8.86	30.26	49.06	66.74
crane [8]	0	59	874	2255	1892	1837	3721	0.00	0.55	8.77	29.97	47.76	65.03
tares [10]	1	100	842	2182	1860	1836	3817	0.01	**0.95**	8.87	29.38	46.86	64.12
lares [10]	1	85	832	2193	1818	1887	3822	0.01	0.81	8.63	29.24	46.33	64.07
rales [10]	1	81	809	2230	1798	1902	3817	0.01	0.77	8.37	29.33	46.23	64.11
rates [10]	1	88	867	2144	1855	1857	3826	0.01	0.84	**8.99**	29.14	46.58	64.04
cares [10]	1	78	818	2159	1866	1909	3807	0.01	0.74	8.43	28.73	46.27	64.22
slate [16]	0	60	829	2244	1939	1866	3700	0.00	0.56	8.35	29.44	47.67	65.21
reais [16]	1	68	823	2060	1808	1953	3925	0.01	0.65	8.39	27.75	44.75	63.11
arose [16]	0	53	779	2055	1893	1994	3864	0.00	0.50	7.82	27.14	44.93	63.67
raise [16, 17]	0	59	791	2074	1852	2020	3842	0.00	0.55	7.99	27.49	44.90	63.89
raise clout	0	15	932	2525	1992	1781	3393	0.00	0.14	8.90	**32.64**	51.37	68.11
raise clout nymph	0	15	557	2841	2762	1397	3066	0.00	0.14	5.38	32.09	**58.05**	71.18

TABLE IV: Simulation Results on the *Wordle Target Data Set*

Seed Words	Count							Cumulative Percentage					
	1	2	3	4	5	6	X	1	2	3	4	5	6
No seed [14]	0	15	241	680	542	552	279	0.00	0.65	11.09	40.54	64.01	87.92
sport chewy admix flunk [6]	1	7	180	431	1053	523	114	0.04	0.34	8.14	26.81	72.41	95.06
soare [7, 15]	0	18	241	663	557	538	292	0.00	0.78	11.22	39.93	64.05	87.35
adept clamp plaid [8]	1	8	80	187	719	764	550	0.04	0.39	3.85	11.95	43.09	76.18
slice [8]	1	17	254	724	557	525	231	0.04	0.78	11.78	43.14	67.26	90.00
tried [8]	1	22	233	719	610	483	241	0.04	0.99	11.08	42.22	68.64	89.56
crane [8]	1	30	267	723	537	493	258	0.04	**1.34**	12.90	44.21	67.47	88.82
tares [10]	0	6	242	684	570	550	257	0.00	0.26	10.74	40.36	65.05	88.87
lares [10]	0	10	222	687	559	554	277	0.00	0.43	10.04	39.79	64.00	87.99
rales [10]	0	12	222	694	577	530	274	0.00	0.52	10.13	40.19	65.18	88.13
rates [10]	0	11	232	673	580	548	265	0.00	0.48	10.53	39.68	64.80	88.53
cares [10]	0	7	262	671	575	543	251	0.00	0.30	11.65	40.71	65.61	89.13
slate [16]	1	18	273	675	567	504	271	0.04	0.82	12.64	41.87	66.43	88.26
reais [16]	0	19	224	642	595	536	293	0.00	0.82	10.52	38.32	64.09	87.30
arose [16]	1	18	217	685	583	525	280	0.04	0.82	10.22	39.89	65.14	87.88
raise [16, 17]	1	16	245	677	552	556	262	0.04	0.73	11.34	40.66	64.57	88.65
raise clout	1	6	342	767	575	455	163	0.04	0.30	**15.11**	48.33	73.23	92.94
raise clout nymph	1	6	203	934	731	354	**80**	0.04	0.30	9.09	**49.54**	**81.20**	**96.53**

words and the number of possible solutions have become less than the number of remaining tries, the algorithm will revert to the best possible suggested word from that point onward instead of attempting the remaining seed words. Given that the maximum number of seed words per-set is four [6] and at least one try has to be entered to start the process, this alternate feature may come into play only on positions $2-4$.

Further, it was observed that in the cases where their simulation discovers four correct characters in an early stage, it gets caught in a plateau where it blindly guesses between multiple equally evaluated answers, which may result in failure. For example, when the target word is `wreak`, the simulation ends up with `-reak` at depth 4. But then it has to select between almost equally valued `freak`, `break`, and `wreak` but only has 2 chances remaining. To mitigate these type of failures, we introduced a heuristic where the simulation first checks if it has already obtained four correct characters. Next it checks to see if the number of remaining possible guesses is larger than the remaining depth. If that is the case, the simulation obtains the unique characters from the remaining possible guesses and attempts an alternate word to clearly eliminate doubt. In the above example, it would extract the characters `f`, `b`, and `w`. Then it will try the word `befit` as its next guess. This will either confirm `f` or `b` to be the correct choice or eliminate them. In this example, the only remaining choice is `w`. But if this was not the case, this could iterate to eliminate more characters. But given the limited depth, we found a single round is all we can afford to sacrifice for this heuristic. The effectiveness of this addition can be seen in the ablation study shown in Fig. 5.

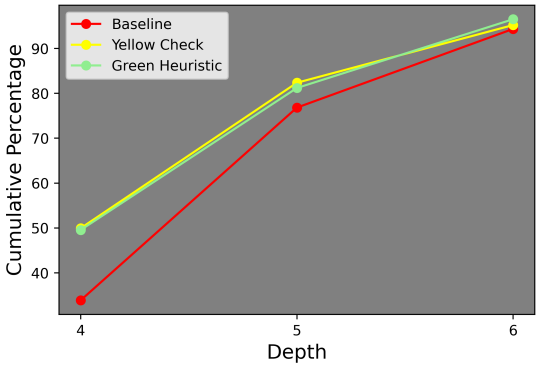


Fig. 5: Ablation study conduced for the *Wordle Target Data Set* using the best seed words `['raise', 'clout', 'nymph']`. We show the *Cumulative Percentage* only at depths $4-6$ in order to be able to see the small differences. The red baseline shows the values for the unaltered solver [14].

The configuration where all three of our seed words are used, show the lowest number of failures for two data sets (refer the X columns). This includes the *Wordle Target Data Set* which represents the possible words a player would encounter in the game for the foreseeable future. Further, the *Cumulative Percentage* columns show how our seed words bring about the ability to solve the *Worldle*s in a reliable manner. This culminates in our three word configuration achieving the highest *Cumulative Percentage* at the depth 4 for the *Wordle Target Data Set* and maintaining the lead till the designated maximum depth of 6.

VI. Conclusion

The objective of this work was to derive the set of 3 optimum starting words for *wordle* covering 15 different characters and ordered in the descending order of significance. We succeeded in that target by discovering the words `['raise', 'clout', 'nymph']`. Our first seed word, `raise`, is as same as that suggested by Groux [16] and Neil [17]. We propose a three seed word strategy and show that our seed words have superior performance to the existing single word methods as well as multi-word methods [6, 8], some of which have 4 seed words. Our proposed seed word set obtain the highest *Cumulative Percentage* within the allocated depth among all the examined methods across all tested data sets. Especially note how our words consistently outperform all the other methods in Table IV which contain the words that have and will be put as daily target words by *Wordle*. Hence, it can be concluded that the seed words proposed in this study, to be the optimal starting strategy for *Wordle* to get the answer as early as possible.

References

[1] J. Wardle, "Wordle - A daily word game," https://bit.ly/Wordle_, 2022, [Online; accessed 07-February-2022].

[2] P. Lok, "Reflecting on medical school while playing wordle," *BMJ*, vol. 376, 2022.

[3] M. Tracy, "The New York Times Buys Wordle," https://nyti.ms/3IxaCAB, 2022, [Online; accessed 01-March-2022].

[4] M. B. Short, "Winning wordle wisely," *arXiv preprint arXiv:2202.02148*, 2022.

[5] B. J. Anderson and J. G. Meyer, "Finding the optimal human strategy for wordle using maximum correct letter probabilities and reinforcement learning," *arXiv preprint arXiv:2202.00557*, 2022.

[6] T. Kandabada, "A Wordle Hack," https://bit.ly/3HFcp5Y, 2022, [Online; accessed 09-February-2022].

[7] D. Sidhu, "Wordle – the best word to start the game, according to a language researcher," https://bit.ly/3GrLxoQ, 2022, [Online; accessed 07-February-2022].

[8] D. Horstmeyer, "Want to master Wordle? Here's the best strategy for your first guess," https://bit.ly/3Gynfta, 2022, [Online; accessed 07-February-2022].

[9] U. Bram and N. Cardin, "The Two Best Ways to Win at Wordle," https://bit.ly/3gLq2oJ, 2022, [Online; accessed 07-February-2022].

[10] M. Butterfield, "Science has determined the worst Wordle starting word," https://bit.ly/3J9FOWj, 2022, [Online; accessed 07-February-2022].

[11] N. D. de Silva, S. M. Weerawarana, and A. S. Perera, "Enabling effective synoptic assessment via algorithmic constitution of review panels," in *Proceedings of 2013 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*. IEEE, 2013, pp. 776–781.

[12] B. Keating, "The frequency of the letters of the alphabet in English," https://bit.ly/3ow5Pr5, [Online; accessed 07-February-2022].

[13] T. Wall and L. Smithline, "English Letter Frequency (based on a sample of 40,000 words)," https://bit.ly/3sgRgIQ, [Online; accessed 07-February-2022].

[14] M. Petersen, "Solving Wordle Puzzles with Basic Python," https://bit.ly/3M7Om2w, 2022, [Online; accessed 28-February-2022].

[15] B. Fan, "The Best Starting Word in WORDLE," https://bit.ly/3K6Hlx7, 2022, [Online; accessed 28-February-2022].

[16] C. Groux, "The 20 best Wordle starting words, according to science," https://bit.ly/3owlqH8, 2022, [Online; accessed 07-February-2022].

[17] T. Neil, "Ruining the fun: a Wordle auto-solver," https://bit.ly/34gRmZj, 2022, [Online; accessed 07-February-2022].