USAGE OF INFORMATION ENTROPY IN SOLVING THE WORDLEGAME

Bsc. Shkodran Hasani[1], Cand. PhD Besart Prebreza[1*], Dr.Sc Arianit Krypa[1*], Cand.PhD Rrezart Prebreza[1*]

E-mail: *shkodran.hasani@universum-ks.org, besart.prebreza@universum-ks.org, arianit.krypa@universum-ks.org, rrezart.prebreza@universum-ks.org*

[*] Corresponding author

[1] *Computer Science, UNI-Universum International College, Pristine, Kosovo*

Keywords: Information Entropy, Probability, Frequency analysis, Wordle, Wordle Solver, Javascript

Received: November 21, 2021

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquipex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eufugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Povzetek: Kratek povzetek povzetka v slovenščini.

1. INTRODUCTION

Wordle is a web-based game created and developed by software engineer Josh Wardle and published by *The New York Times Company* in early 2022. Conceptually Wordle is similar to the 1970 game Mastermind (Machkovech , 2022), but which uses letters (*Wordle*) instead of colors (*Mastermind*). Wordle has a single daily solution where all players have six chances to find a five-letter word which is selected as the answer, after each guess theplayer is provided with clues indicating when the letters match or occupy the correct position.

Although the rules are easy to understand, and no prior knowledge is needed, the main challenge lies in reaching the final answer in as few trials as possible. This has led many people around the world to try to build different algorithms and techniques with the sole purpose of finding the final answer in as few trials as possible.

Taking into account the facts mentioned above, in this paper we will mainly talk about information entropy and its use in the algorithm for solving Wordle, in this way we will be able to reinforce our knowledge in some mathematical concepts in a more comprehensive and fun way. Finally, we will cover Wordle's algorithm from a programming perspective (projects made in JavaScript [1]), where we will understand the function that each piece of code performs, the results it provides us, and testing the performance of the algorithm in comparison to other algorithms used to solve Wordle.

2. LITERATURE REVIEW

When building the algorithm for solving Wordle, we used the formula for measuring information entropy, developed by *Claude Shannon.* During the 1940s Shannon was trying to develop a way to quantitatively measure information, its compression [2], and the impact of the deterministic nature of language on information entropy. In 1948, he published his findings in a book called "A Mathematical Theory of Communication*".*

Later in 1963, the book was reprinted together with Warren Weaver's article called "*The Mathematical Theory of Communication"* on which this paper is based.

Information theory

Communication is the process where through sounds, signs or actions we transmit information from the sender to the receiver. This information can be reflected in different forms (through speech, musical notes, pictures, etc.) by different people, with the main purpose of sharing information [7].

Just like using scales to measure the weight of different things, information regardless of the communication method can be measured using a type of measure, which is entropy, and the way of its quantitative representation is realized using the unit of measure bit (abbreviatedfrom the word "binary digit"), where one bit represents the information we can extract from a question that accepts only two possible values as an answer: yes or no. [7].

Information

An example through which we can understand how we can measure information would be flipping a coin. Suppose person **A** flips a coin, how many **yes or no questions** must person **B** be asked in order to find the result of the coin flip?

Since the coin has only two possible outcomes: "heads" or "tails", person **B** can find the result by asking only one question: *Is the result "head"?* If the answer of person **A** is "*yes*", then he has come to the conclusion that the result was "head", while if the answer is "*no*", then the result was "tail". From this example we conclude that the coin flip contains one bit of information.

But what if instead of flipping a coin it was rolling a dice, how many questions are needed to correctly identify the winning number?

Let's assume that person **A** got the number "5" when rolling the dice, since we have six possible values as a result, then in order to ask the minimum number of questions, person B **needs** to halve the number of outcomes during each question. Person **B** asks: *Is the result greater than "3"* ? The answer from person **A** will be " *no* ", this leaves us with only three possible outcomes (*4, 5 or 6*) the next question should be: *Is the result "4"* ? the answer from person **A** will be " *no* ", which leaves us with only two possibilities (*5 or 6*), where with the next question person *B* can undoubtedly find the correct result. To find the number "5" three questions are needed, but when throwing the dice we have cases when we may need only two questions (example: *finding the number "4")* . Measuring information in such cases would be easier by referring to a formula.

From the observation of different events we come to the conclusion that if we have two possible solutions, one question is needed to find the answer, if we have four possible solutions two questions are needed, for eight possible solutions three questions are needed and so on.

For:

$$k = 2, 4, 8, 16 \ldots \Rightarrow I = 1, 2, 3, 4 \ldots$$

From where we derive the formula:

$$k = 2^I$$

$$I = log_2 k \qquad (1)$$

I - information (number of questions)

k - number of possible outcomes (space of posibilities)

Therefore we can say that the throw of the dice contains $log_2 6 \approx 2.6$bits of information.

Entropy of information

But does an event produce the same information if the probabilities between the possible events are not equal? Suppose we have two applications, each generating astring of characters consisting of 4 letters (*example* : A, B, C, and D) [4]. Application **1** generates letters completely randomly $(25\%\ A,\ 25\%\ B,\ 25\%\ C,\ 25\%\ D)$, while application **2** generates letters according to probability $(50\%\ A,\ 12.5\%\ B,\ 12.5\%\ C,\ 25\%\ D)$, to understand how many bits of information each letter contains, we use the above method (halving the probability during each question):

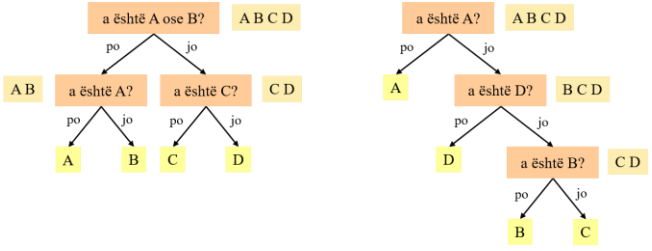

Figure 1. Visual representation of measuring information

In the case of application **1,** we can say that each letter contains exactly two bits of information, since a minimum of two questions are needed to identify each letter, but in application **2**, the letter A has one bit of information, the letter D has two bits of information,while the letters B and C have three bits of information each.

If we calculate the average:

$$I = p_a \cdot I_a + p_b \cdot I_b + p_c \cdot I_c + p_d \cdot I_d$$

$$I = 0.5 \cdot 1 + 0.125 \cdot 3 + 0.125 \cdot 3 + 0.25 \cdot 2$$

$$I = 1.75\ bit$$

p – probability

From this example we understand that the less random the letter generation is, the less information a given event contains. From this conclusion Shannon gave his formula for calculating the entropy of information or as he called it: "the freedom of choice" [7].

$$H = p_1 \cdot I_1 + p_2 \cdot I_2 + \cdots + p_n \cdot I_n$$
$$H = \sum_i p_i \cdot I_i$$
$$H = \sum_i p_i \cdot \log_2(1/p_i)$$

$$H = -\sum_i p_i \cdot \log_2 p_i$$

H – information entropy (*bit*).

Entropy of language

If we analyze the frequency of the letters of a language we can notice that the selection of letters for theformation of words is not completely random, since not all letters are used in the same frequency in the sentence,a concrete example can be seen by analyzing the English language (chart below).

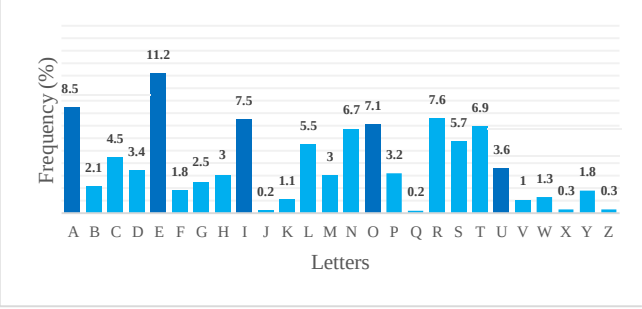

Figure 2. Frequency of letters in the English language [8]

By reading the graph, we see that the most used letter in the English language is "E", while "Q" is the least used letter, or from a frequency perspective; the letter E occurs 56 times more in sentences than the letter Q. From another point of view, we see that the use of vowels is higher than that of consonants. This is due to the fact thatdespite the small number of vowels (only five) after every one or two consonants the next letter in the sentence will most likely be a vowel.

We encounter such things with letters and words in particular, *for example:* The possibility that a word that begins with the letter *j* is followed by the letter: *c, f, g, j, q, v, w, x* or *z* ; is almost zero, since there are only a few such words that meet this condition, it is also very unusual that after the word "tradition" come the words "pink" since their meaning is not related.[7]

From the facts mentioned above, we can say that the selection of letters and words in forming sentences and messages is not completely independent, since the language itself limits us in our selection.

Game rules

At the beginning of the game, the player finds himself in front of a table 5×6 with empty cells, in which he will place the words he sees as correct, after each attempt each letter is marked with green, yellow or gray color, the green cell means that the letter placed in it is correct and in the correct position, the yellow cell means that the corresponding letter is found in the final answer but in a different position, while the gray cell means that such letters are not part of the final answer.

Also words that have more than one letter of the same will be marked in green or yellow color only if the final answer also has more than one such letter, otherwise only the first letter of this type will be marked with the color green/yellow while other letters will be marked in gray.

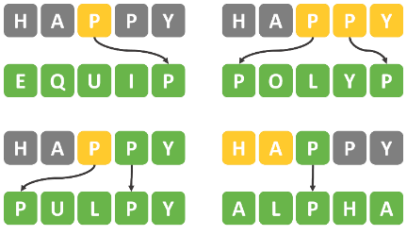

Figure 3. Color marking of the letters of the word "Happy" for different answers

The game contains the possibility of changing the view from bright to dark and vice versa (*light mode / dark mode*) as well as offers the possibility of using the high contrast color set, for people with daltonism (color blindness) where the green color is replaced by the orange color, while the yellow color is replaced by the blue color.

The game contains 12972 words of which 2315 are possible answers, the player during each guess can try one of these 12972 words, while if the player chooses the hard mode option then the player is *obliged* to use all letters marked with green and yellow colors detected during preliminary guesses in its subsequent answers.

Applying entropy to Wordle

So far we have discussed information entropy and the Wordle game, but we have not yet discussed how we can incorporate information entropy into solving Wordle, and how efficient this method is.

Measuring word information

Earlier we mentioned that the unit for measuring information is *the bit* , where we also mentioned the formula *equation* (1) which means that if an event halves the total number of possible cases, then we say that there is a bit of information. In our case the total number of cases represents the list of 12972 words and it turns out that about half of these words contain the letter "S",therefore this event provides us with about a bit of

information, also a quarter of the words contain the letter "T" which means that such events have two bits of information[6].

Unlike probability, where the final result is the product of the probability of all events, the information on the other side has the result of the sum, *example:* if the word selected by the player contains the letter S and T, then the probability of the event is:

$$p_{s\&t} = p_s \cdot p_t = (1/2) \cdot (1/4) = 1/8$$

While the information it provides is:

$$I_{s\&t} = I_s + I_t = 1 + 2 = 3\ bit$$

In this way, the information contained in a word is calculated by iterating[3] to each letter in the corresponding word, we find how many bits of information each one of them holds, and finally we calculate the total amount.

Using patterns[4]

Another feature of the game that helps us find the optimal word are the patterns that show us which words are found or not in the final answer. By marking each letter of the corresponding word in green, yellow or gray,we can precisely eliminate from the list all the words thatdo not adhere to the given pattern, thus reducing thepossibility of selecting the unwanted word.

To analyze the quality of the word selected by us, when it is known that the pattern of this word depends on the final answer, it is through the generation of all possible patterns and the calculation of the number of words that fits in them, Since a letter can be marked with one of three colors (green, yellow or gray) and a word contains five letters, then in total we have $3^5 = 243$ patterns. If our selected word is "*focus*" then the distribution of words in the given patterns will look like:

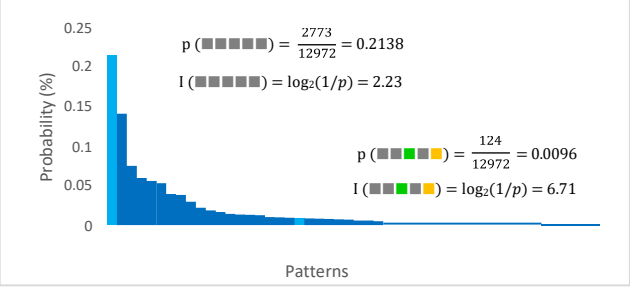

Figure 4. Presentation of the probability for each pattern of the word "focus"

In the chart above, we've sorted the patterns from left to right by the number of words that match them, and we've only shown the first 50 patterns. In the word "*focus*" in the most frequent case (21% of cases) when all the letters are marked in gray, the information we get is only 2.23 bits, which means that at least from 12972 possible words we reduce the possibilities space to 2773 words, while there are cases when we get 6.71 bits of information from this word, this happens when the letter "*C*" is marked in green and "*S*" in yellow, only 124 words fit this pattern, but we only encounter it 1% of cases.

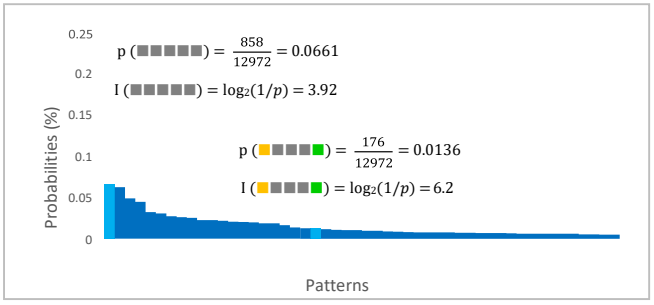

Figure 5. Presentation of the probability for each pattern of the word "tares"

Another word that provides us with more information is the word "*tares*", if we look at its graph we can see that the distribution of words in the pattern is greater, since even in the worst case (when all the letters are marked gray) the minimum information we get is 3.92 bits (reducing the number of possible words to 858), but this pattern is encountered in only 7% of cases, which means that usually we get more information than that.

From the graphs above, we see that the more information a certain word contains, the lower its probability is, therefore, for the selection of the optimal word, we refer to the formula of equation (2) where we calculate the product between the probability of a corresponding pattern and *the* information provided by that pattern, their total sum represents the average value of the information provided by that word.

By calculating the entropy for the words *focus* and *tares*, we see that the word focus provides an average of 4.54 *bits* of information, while the word tares provides 6.19 bits. From these results we see that using the word tares instead of the word focus reduces the list of words by about three times more.

To understand the limitation that language made us in the selection of words, we can compare the entropy of the word with the highest value (tares - 6.19) with the maximum entropy that a word can have:

$\log_2 3^5 \approx 7.92\ bit$ and the number of possible words from ($26^5 =$ 11,881,376 words (23.5 bits) to 12972 words (13.66 bits).

[3] Iteration *means* executing a piece of code repeatedly until a certain condition is met

[4]Pattern/s (*here*) means the way of presentation of the result depending on the marking of the letters in green, yellow or gray

3. APPLICATION

Building the Wordle algorithm

Generating solutions to the Wordle game requires discovering strategies that lead to the final answer in as few guesses as possible. Eliminating as many words as possible and analyzing the remaining words are the keys to these strategies.

During the construction of such algorithms, we must take into account several main elements such as: the accuracy of the algorithm, the time complexity and the space it occupies in the memory. Since our game is about selecting words from a pre-written list, some of the optimal techniques are:

- Filtering incorrect words
- Using predefine words
- Letter frequency analysis
- Analyzing the frequency of letters and their positions in words
- Calculating information entropy
- Calculating information entropy (using the answer list)

The main reason why I have used Javascript in my project work is the ease of access. Javascript is a web scripting language [5] , so there is no need to install additional software since the code is executed in your browser, while applications like text-editor (such as Notepad) can be used to write the code.

In this section, we'll look at building code for the *Calculating information entropy* and *Calculating information entropy (using the answer list)* algorithms.

Application structure

The files will be divided into several main folders:

- **data** where the list of words and answers will be stored (words.js, answers.js),
- **scripts** where the code for building the game (index.js, game.js) and the code for building the algorithms will be stored,
- **testing** code for testing the performance of algorithms
- **styles** code for designing the web page (.css files)
- **index.html** main file through which the result of the application is displayed

Calculating information entropy

This method is based on the selection of words that contains the most information and at the same time reducing the list of possible words to its minimum. The algorithm starts by calculating the entropy of each word for each possible pattern, and finds their sum, the word with the highest value is selected as an valid answer.

```
1  import      {      possibleWords      }      from
2  './../data/words.js';
3  import { matchedWords } from './filter.js';
4  let wordsLength = 5;
5  let bestWord;
6
7  let patterns = [];
   const numberOfPatterns = Math.pow(3, wordsLength);
```

To begin with, we imported the list `possibleWords`, which contains all possible words, and the list `matchedWords`, which will store the words that meet the condition of the pre-discovered patterns, we declared the variable `wordsLength` which holds the value of the number of letters in a word (five) and the variable `bestWord` in which to store the optimal word for the answer.

We have also declared the `patterns` list which will carry the value of all possible patterns of a word (numbers 2, 1 or 0 will be used for the potential marking of green, yellow or gray letters). We have also created the `numberOfPatterns` variable which shows the possible number of patterns: 243 [6].

```
8  for(let i = 0; i < numberOfPatterns; i++) {
9    patterns[i] = [];
10   let                    temp                    =
11 i.toString(3).padStart(wordsLength,0);
12   for(let j = 0; j < wordsLength; j++) {
13     patterns[i][j] = parseInt(temp[j]);
14   }
15 }
16
17 function informationEntropy() {
79   ... // algorithm code
   }
```

The generation of these patterns is done automatically using the `toString()` method, which converts a decimal number (with base 10) into a number with base 3(since it has the number 3 as a parameter). In parallelwith the generation of numbers, the `padStart()`method is used, which has the task of converting each generation of the pattern into a five-digit value by adding zero to the beginning of the number as needed.

[5] (*Eng. Scripting language*) means programming language which interprets and executes commands one by one

[6] Three marking options: green, yellow, gray - for every five letters in a word; $3^5 = 243$pattern in total

```
let matches = matchedWords.length;
let entropyValue = 0;
let entropyArray = [];
for(let h = 0; h < possibleWords.length; h++) {
  for(let i = 0; i < numberOfPatterns; i++) {
    for(let j = 0; j < matchedWords.length; j++) {
      for(let k = 0; k < wordsLength; k++) {
        ... // if statements
      }
    }
    ... // reset matches
  }
  ... // reset entropyValue
}
```

inside `informationEntropy()` method we have declared the variable `matches` which will contain the number of words that match a certain pattern, the `entropyValue` variable which will contain the average entropy value of a given word and the `entropyArray` list in which the entropy values of all words will bestored. This is achieved by iterating over four nestedloops.

```
if(patterns[i][k] === 2) {
  if(possibleWords[h][k]!==matchedWords[j][k]) {
    matches--;
    break;
  }
}
```

If the specified pattern has the value two then every word is eliminated from the `matchedWords` list which does not contain the specified letter in current position (since the word; `possibleWords[h]` in which the entropy is being calculated contains this letter in this position).

```
else if(patterns[i][k] === 1) {

if(!matchedWords[j].includes(possibleWords[h][k])
|| possibleWords[h][k] === matchedWords[j][k]) {
    matches--;
    break;
  }
}
```

If the pattern has the value one then every word that does not contain the given letter or even the words that contain the letter in the same position is eliminated (because the word `possibleWords[h]` contains this letter but in a different position).

```
else {
  if(matchedWords[j].includes(possibleWords[h][k]))
{
    matches--;
    break;
  }
}
```

If pattern is zero, then remove all words containing this er let(since the word `possibleWords[h]` does not ntain cothis letter at all).

```
if(matches > 0) {
  let probability = matches / matchedWords.length;
  let information = Math.log2(matchedWords.length
/ matches);
  entropyValue += probability * information;
}
matches = matchedWords.length;
```

After finishing the elimination of the words that do not match, we start with the entropy calculation for each possible pattern, if the value of the words that match this pattern is at least one `(matches > 0)`. The entropy of the word is obtained by calculating the total sum of the product between the probability of a pattern and the information that this pattern provides (see *equation* (2)). Tlen, the variable `matches` is reset for entropy lculation caof the next word.

```
entropyValue                                    =
parseFloat(entropyValue.toFixed(2));
entropyArray[h] = entropyValue;
entropyValue = 0;
```

Using the `toFixed()` method the calculated entropy values are rounded to two decimal places and stored in the `entropyArray` list, as well as the `entropyValue` value is reset at the end, as it will be used to store the entropy value of the next word.

```
indexArray = [];
for(let i = 0; i < possibleWords.length; i++) {
  if(matchedWords.includes(possibleWords[i])) {
    indexArray.push(i);
  }
}
let maxValue = entropyArray[0];
bestWord = possibleWords[0];
let tempIndex = 0;
```

After we have calculated the information entropy for all the words, we need to find the maximum value of entropy. In advance, for all the words that can be

potential answers of the game (`matchedWords`), westore their indexes in a list called `indexArray`.

```
for(let i = 0; i < entropyArray.length; i++) {
  if(entropyArray[i] > maxValue) {
    maxValue = entropyArray[i];
    bestWord = possibleWords[i];
    tempIndex = i;
  }
  else if(entropyArray[i] === maxValue) {
    if(indexArr.includes(i) &&
    !indexArr.includes(tempIndex)) {
      maxValue = entropyArray[i];
      bestWord = possibleWords[i];
      tempIndex = i;
    }
  }
}
```

By iterating through the entire list we find the maximum value and store it in the `maxValue` variable and the corresponding word in the `bestWord` variable. An extra step we should take, also the reason for using the `indexArray` list is the detection of equal values. Incase two or more words have the same entropy values, then the maximum value is selected based on which wordis found in the list of possible answers.

Calculating information entropy (using the answerlist*)*

A more efficient version of the algorithm is using the list of answers instead of all possible words, thus increasing the speed and accuracy of the algorithm

```
// informationEntropy code
let matchedWords = [...possibleWords];

// informationEntropyAdvance code
import { possibleAnswers }
from './../data/answers.js';
let matchedWords = [...possibleAnswers];
```

The implementation of the algorithm is also the same, it is enough to import the list of possible answers (`possibleAnswers`) and create the list `matchedWords` which has this same list as a source, unlike the previous algorithm (where `matchedWords`has been the source for the list `possibleWords`).

4. DATA ANALYSIS AND DISCUSSION

In order to analyze the performance of these algorithms, we have built several pieces of code that test the algorithms for all 2315 possible words as answers, I have shown the obtained data below in the form of graphs and tables.

In this part we will present the algorithm analysis Using information entropy and Using information entropy (using answer list).

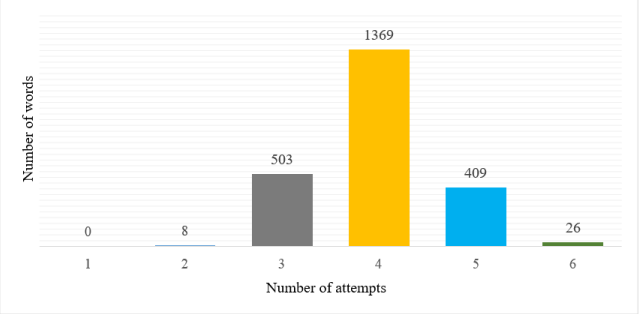

Figure 5: Test plot of the "Using information entropy" algorithm

From the graph above, we see that the algorithm has found the correct answer in the most frequent case in 4 attempts (1369 cases), it has never managed to find the correct answer in the first case, while the maximum number of cases required is 6 attempts (26 cases).

Has won	Lost	Average	TOTAL
2315	0	3.97	2315

Table 1. Results from testing the algorithm "Using information entropy"

From the table we conclude that the use of information entropy as an average has achieved a significant improvement with 3.97 attempts for answers, has not failed in any case in finding the final answer (0 cases, or expressed as a percentage of 0%) and has completed successfully in 2315 cases or 100%.

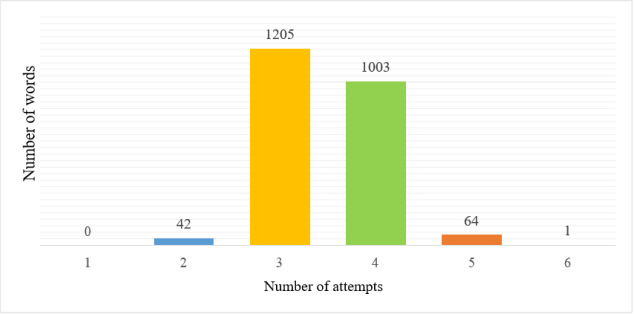

Figure 6: Test plot of the algorithm "Using information entropy (using response list)

From this graph we see that the use of the list of answers by the algorithm has given more accurate results by reducing the number of attempts in the most frequent case to find the answer to 3 (1205 cases), also it has never managed to find the answer correct in the first case, while the maximum number of cases needed to find the answer is 6 attempts (1 case).

Has won	Lost	Average	TOTAL
2315	0	3.47	2315

Table 2. Results from testing the algorithm "Using information entropy (using the list of answers)"

From the table we see that the use of the answer list in the algorithm that uses entropy as an average has performed in 3.47 attempts for answers, an improvement on the average, while it has never failed (0 cases or 0%) and ended successfully (2315 cases or 100%).

5. CONCLUSIONS AND RECOMMENDATIONS

By testing the algorithms for finding the optimal answer for the Wordle game, we can see in detail the advantages and disadvantages that each algorithm offers in terms of performance, accuracy, and memory allocation.

The algorithm for filtering incorrect words is among the easiest algorithms to build, has good time performance and is an integral part of other algorithms, but it does not perform well in accuracy. The use of predefine words is mostly easy to build, it alsooffers good performance in terms of time (both duringthe game and during testing) since the selection of words is done at the beginning of the game and does not change until its completion. There are no major improvements over the previous algorithm in terms of accuracy, but it significantly reduces the number of cases of failure to find the correct answer within 6 possibilities.

The letter frequency analysis provides relatively goodperformance in terms of time, but offers much higher accuracy in finding answers, while the algorithm for analyzing the frequency of letters in the corresponding positions performs almost the same in terms of time,while providing even higher accuracy.

Using information entropy is more complicated to construct and consumes more time and processing power,but it provides the most accurate results and never fails tofind the answer, also using a list of answers increases accuracy even more of the algorithm and significantly improves the time performance.

As a final selection for solving the Wordle game, it would be to use information entropy since it has the highest accuracy, also the complexity of the algorithm does not present a problem in terms of time.

6. REFERENCE

[1] Cao, S., & Dahir, I. (2022, January 16). How Wordle Became The Internet's Omicron Pastime. Retrieved from BuzzFeed News: https://www.buzzfeednews.com/article/stefficao/how-wordle-went-viral-strategy

[2] Clark, M. (2022, January 24). Twitter suspends Wordle-ruining bot. Retrieved from The Verge: https://www.theverge.com/2022/1/24/22899339/wordle-twitter-spoilers-banned-word-puzzle-answers

[3] Greenbaum, A., & Byrd, M. (2022, July 13). 20 Best Wordle Clones For Your Daily Game Needs. Retrieved from Den of Geek: https://www.denofgeek.com/games/best-daily-games-wordle-clones-spin-offs/

[4] Khan Academy. (2014, April 28). Information entropy. Retrieved from https://www.khanacademy.org/computing/computer-science/informationtheory/moderninfotheory/v/information-entropy.

[5] Machkovech, S. (2022, March 25). Wordle creator describes game's rise, says NYT sale was "a way to walk away". Retrieved from arsTechnica: https://arstechnica.com/gaming/2022/03/wordle-creator-describes-games-rise-says-nyt-sale-was-a-way-to-walk-away/

[6] Sanderson, G. (2022, February 6). Solving Wordle using information theory. Retrieved from 3blue1brown: https://www.3blue1brown.com/lessons/wordle

[7] Shannon, C., & Weaver, W. (1963). The Mathematical Theory of Communication. Urbana, United States of America: The University of Illinois Press. Retrieved from https://monoskop.org/images/b/be/Shannon_Claude_E_Weaver_Warren_The_Mathematical_Theory_of_Communication_1963.pdf

[8] University of Notre Dame. (1995). The frequency of the letters of the alphabet in English. Retrieved from Concise Oxford Dictionary:

[9] https://www3.nd.edu/~busiforc/handouts/cryptography/letterfrequencies.html