WORDLE PLAYER PROJECT

Bohan Feng (NUID 001564249)
Olasunkanmi Olayinka (NUID 001512266)

https://github.com/fengb3/INFO6205_final_project

Table of Contents

Introduction about the topic:

Wordle is web-base word game in which the goal is to guess a five-letter word, and you are given 6 different chances to guess the word, each time you make a guess you get some information about how close the word you guessed is close to the actual word. The letters are color coded; grey means the letter doesn't exist in the word, yellow mean the letter exist but it's in the wrong position and green means the letter exist and is in the right position.
This project introduces a software that takes the word guess with each letter's color code and calculate the next best possible guess word using information theory.

Aim of the project:

Create a wordle player that will choose the best words for any given move, according to the principles of entropy, here by optimizing the number of chances it takes you to guess the correct word

Steps to implement solution:

- Compare each pair of words to get a pattern matrix
    - We use a look up for every time we compare 2 words to speed up the process
- Calculate the expected amount of information we can get from each word in a word list
    - If we enter a word to wordle game, the pattern we get can help us exclude half of the words, we can define that we reduce 1 bits of information. For a possibility of a pattern P, we need I inputs, where I satisfy $P = (\frac{1}{2})^I$. After simplifying the formula, we can get $I = -log_2 P$ where I is the amount of information provided by a pattern for a word.
    - The entropy is the expected Information of a word, so we loop though all the pattern for each word in legitimate word list to get the probability of matching another word for a pattern which is $P(pattern)$
    - Then the amount of information provided by each pattern is weighted to obtain the information entropy of a single word.
    - So, the formula to calculate the entropy of a word is

$$E_{word} = \sum_{pattern \in 3^5} P(pattern) * log_2 P(pattern)$$

- Get the word frequency to know how likely a word is to be a common word
    - We use a data set to get the frequency of a word in English language
    - Based on the data set, we sort all the 5 letters words by their frequency
    - The use the sigmoid function to give each word a probability to be common word
    - The sigmoid function gives each word a probability that almost 0 or 1. So it is easier to identify if a word is a common word

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- Combine entropy and probability to give each word a rank as recommendation
    - We basically add 2 values
- Use two steps to get best beginner words
    - For each opening word, we try to find the word with the largest amount of information in the list of valid words generated after they satisfy a certain pattern as the best choice for the second step.
    - Then add the information entropy of the first word and the best second word to get the information entropy of the 2 steps entropy for each opening word.
    - Then we use each word in words list as the answer to test the top 50 entropy opening word to see which opening gives the best score.

Build steps and requirements:

You will need JDK 11 or greater and IntelliJ idea for build the application

1. Download and unzip project into directory of choice
2. Setup project SDK in IntelliJ with Java 11 or greater
3. Run the Main.java file
   Note: this will take time (15 – 40 mins) running the first time to create entropy_map and pattern_matrix, to reduce time you can add files entropy_map.txt and pattern_matrix.txt (this couldn't me pushed to GitHub because of file size limitation) to the data directory

Complete project details and its functionalities:

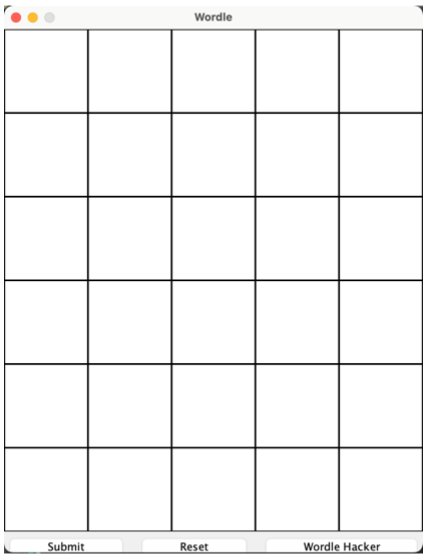
- Start the application by running Main.java
  The Wordle application UI should pop up (you can play regular wordle with this)
- Click on Wordle Hacker button to open the Wordle Player (Hacker) UI
  The Hacker UI should pop up showing Top pick, Entropy, and the Probability
- Play the Wordle application and the Hacker UI is auto populated showing next best possible words sorted with respect to probability.
- You can manually populate the Hacker app with your guess words and select the appropriate color, if you are playing wordle on a different platform or application
- You can run a Benchmark by clicking the benchmark button
  This will run a test on 2309 guess words and returns the average number of chances it takes the Wordle Hacker to get the guess word. It logs how many tries it takes for each guess word on the console and writes the average number of chances into a file data/benchmark_$dateTime.txt

A Simple run of the application

There are two windows in this application.

Each window can work separately or together, so the user can use the hacker to play with the real wordle game. when both of window are opened, the Hacker window will its statics based on the Wordle window.

Wordle app:



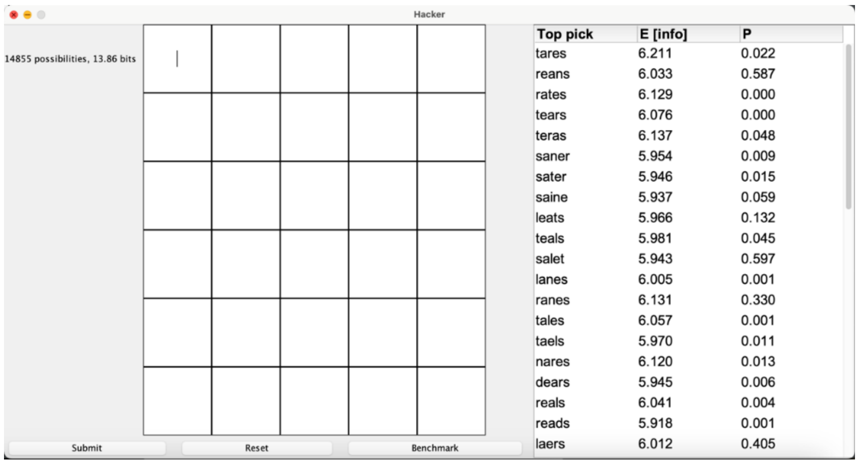
Wordle application is simply wordle game that allow user to play multiply times.

Submit button will check the word's pattern between and move on the the next try.

Reset button will restart the game and set the answer to a new random word

Wordle Hacker button will open a new window named Hacker

Hacker app:

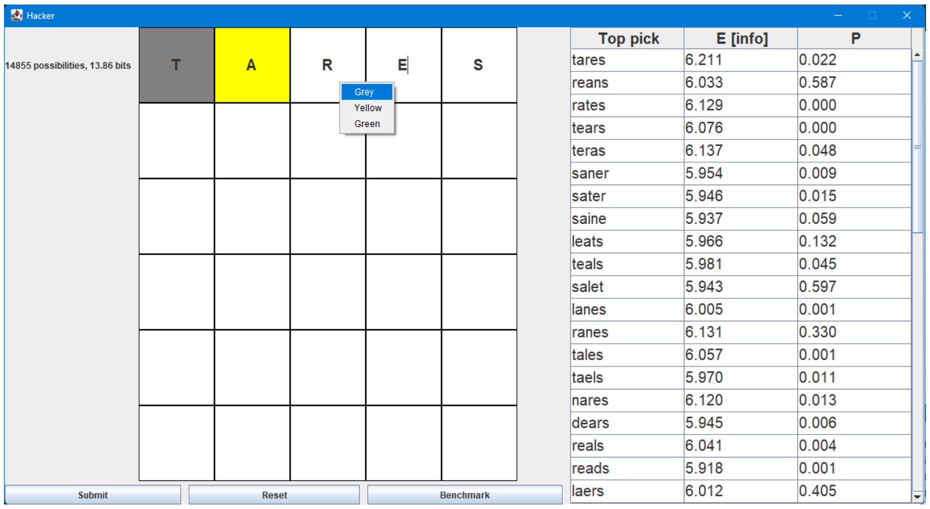


14855 possibilities, 13.86 bits

Top pick	E [info]	P
tares	6.211	0.022
reans	6.033	0.587
rates	6.129	0.000
tears	6.076	0.000
teras	6.137	0.048
saner	5.954	0.009
sater	5.946	0.015
saine	5.937	0.059
leats	5.966	0.132
teals	5.981	0.045
salet	5.943	0.597
lanes	6.005	0.001
ranes	6.131	0.330
tales	6.057	0.001
taels	5.970	0.011
nares	6.120	0.013
dears	5.945	0.006
reals	6.041	0.004
reads	5.918	0.001
laers	6.012	0.405

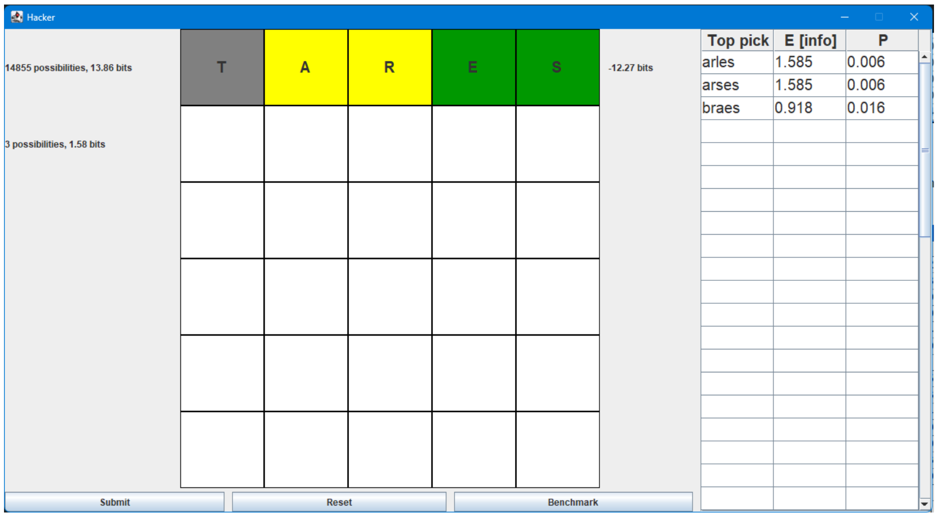
Hacker application displays statics of the wordle game

There is a table at the right side of the Hacker window, which displays the entropy and probability to be a common word in legitimate word list.



Top pick	E [info]	P
tares	6.211	0.022
reans	6.033	0.587
rates	6.129	0.000
tears	6.076	0.000
teras	6.137	0.048
saner	5.954	0.009
sater	5.946	0.015
saine	5.937	0.059
leats	5.966	0.132
teals	5.981	0.045
salet	5.943	0.597
lanes	6.005	0.001
ranes	6.131	0.330
tales	6.057	0.001
taels	5.970	0.011
nares	6.120	0.013
dears	5.945	0.006
reals	6.041	0.004
reads	5.918	0.001
laers	6.012	0.405

Right click the input box in wordle grid will pop up a menu, that allows the user to set the pattern of each guess



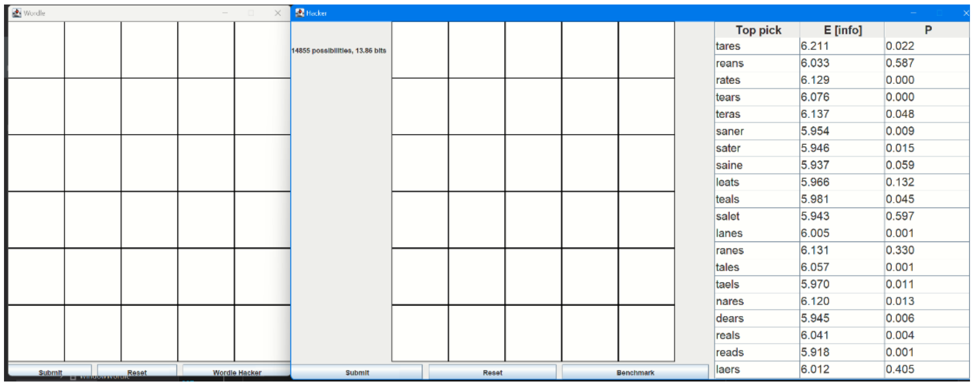
Top pick	E [info]	P
arles	1.585	0.006
arses	1.585	0.006
braes	0.918	0.016

once a valid pattern is set to a word, clicking the submit button will update the legitimate word list and sort them by its entropy.

after each submission, the hacker will display how much word in the legitimate words, and the remaining information in terms of bits at the left of the wordle grid. Also, it displays how much information reduced after each try.

You can open the real wordle game, choose a guess from the legitimate word list, and set the pattern responded by the real wordle, to get recommendation for next try.

Wordle and Hacker work together

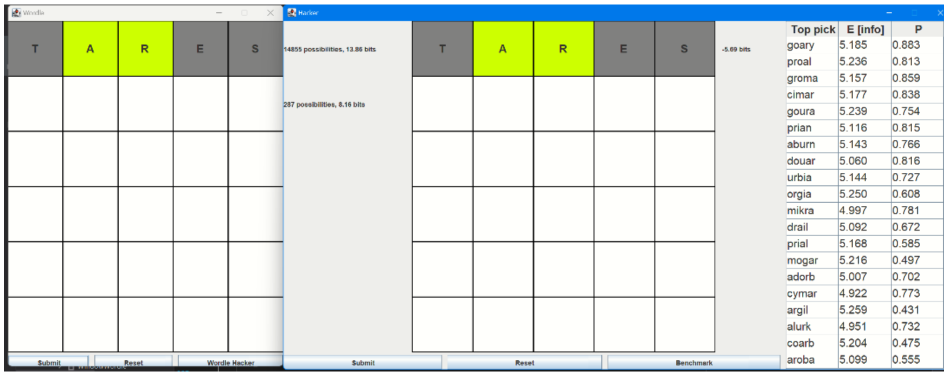


14855 possibilities, 13.86 bits

Top pick	E [info]	P
tares	6.211	0.022
reans	6.033	0.587
rates	6.129	0.000
tears	6.076	0.000
teras	6.137	0.048
saner	5.954	0.009
sater	5.946	0.015
saine	5.937	0.059
leats	5.966	0.132
teals	5.981	0.045
salet	5.943	0.597
lanes	6.005	0.001
ranes	6.131	0.330
tales	6.057	0.001
taels	5.970	0.011
nares	6.120	0.013
dears	5.945	0.006
reals	6.041	0.004
reads	5.918	0.001
laers	6.012	0.405

Steps on playing this game when using Wordle and Hacker

1. enter a word to the Wordle application and click the submit button



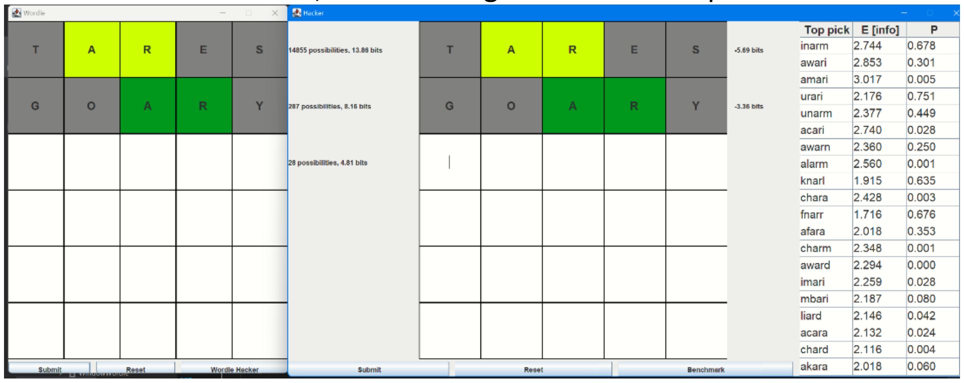
14855 possibilities, 13.86 bits

-5.69 bits

287 possibilities, 8.16 bits

Top pick	E [info]	P
goary	5.185	0.883
proal	5.236	0.813
groma	5.157	0.859
cimar	5.177	0.838
goura	5.239	0.754
prian	5.116	0.815
aburn	5.143	0.766
douar	5.060	0.816
urbia	5.144	0.727
orgia	5.250	0.608
mikra	4.997	0.781
drail	5.092	0.672
prial	5.168	0.585
mogar	5.216	0.497
adorb	5.007	0.702
cymar	4.922	0.773
argil	5.259	0.431
alurk	4.951	0.732
coarb	5.204	0.475
aroba	5.099	0.555

2. The wordle grid on both wordle window and hacker will set to pattern by comparing the wordle answer. Also, the list of legitimate will be updated.



14855 possibilities, 13.86 bits

-5.69 bits

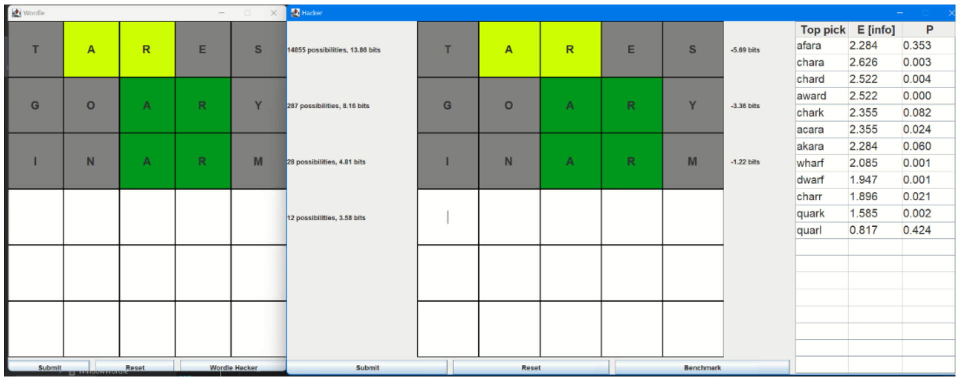
287 possibilities, 8.16 bits

-3.36 bits

26 possibilities, 4.81 bits

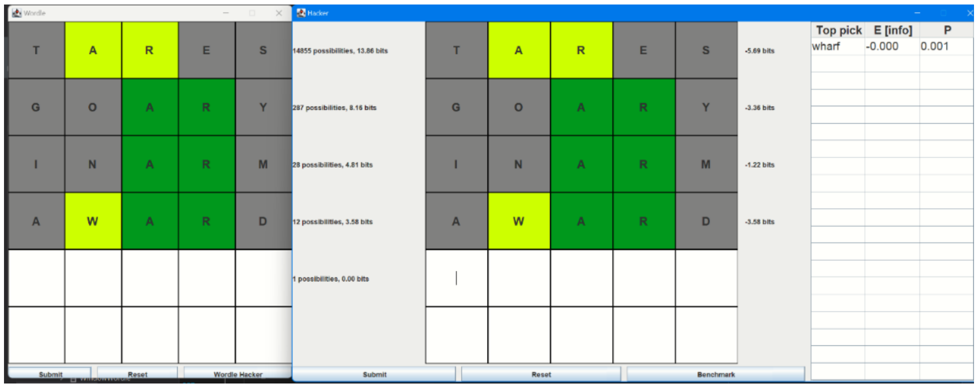
Top pick	E [info]	P
inarm	2.744	0.678
awari	2.853	0.301
amari	3.017	0.005
urari	2.176	0.751
unarm	2.377	0.449
acari	2.740	0.028
awarn	2.360	0.250
alarm	2.560	0.001
knarl	1.915	0.635
chara	2.428	0.003
fnarr	1.716	0.676
afara	2.018	0.353
charm	2.348	0.001
award	2.294	0.000
imari	2.259	0.028
mbari	2.187	0.080
liard	2.146	0.042
acara	2.132	0.024
chard	2.116	0.004
akara	2.018	0.060

3. choose one of the words from Hacker's word list and type into the Wordle window. Then press submit in wordle window to get the pattern and updated word list.



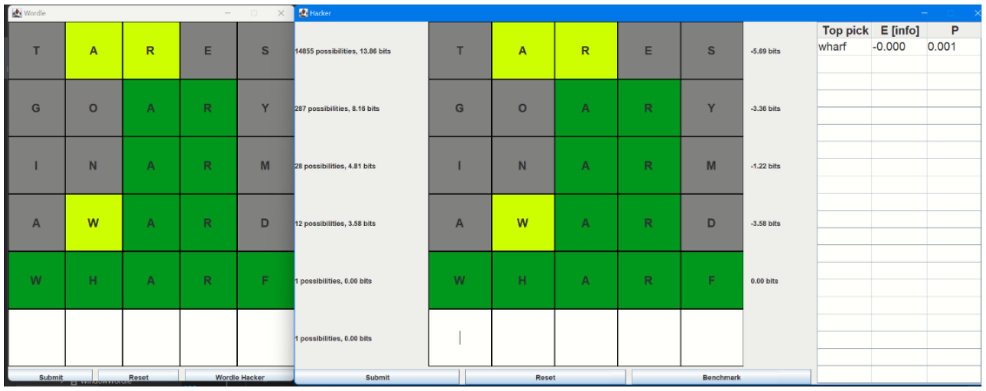
```
14855 possibilities, 13.86 bits   -5.69 bits
287 possibilities, 8.16 bits      -3.36 bits
28 possibilities, 4.81 bits       -1.22 bits
12 possibilities, 3.58 bits
```

Top pick	E [info]	P
afara	2.284	0.353
chara	2.626	0.003
chard	2.522	0.004
award	2.522	0.000
chark	2.355	0.082
acara	2.355	0.024
akara	2.284	0.060
wharf	2.085	0.001
dwarf	1.947	0.001
charr	1.896	0.021
quark	1.585	0.002
quarl	0.817	0.424



```
14855 possibilities, 13.86 bits   -5.69 bits
287 possibilities, 8.16 bits      -3.36 bits
28 possibilities, 4.81 bits       -1.22 bits
12 possibilities, 3.58 bits       -3.58 bits
1 possibilities, 0.00 bits
```

Top pick	E [info]	P
wharf	-0.000	0.001

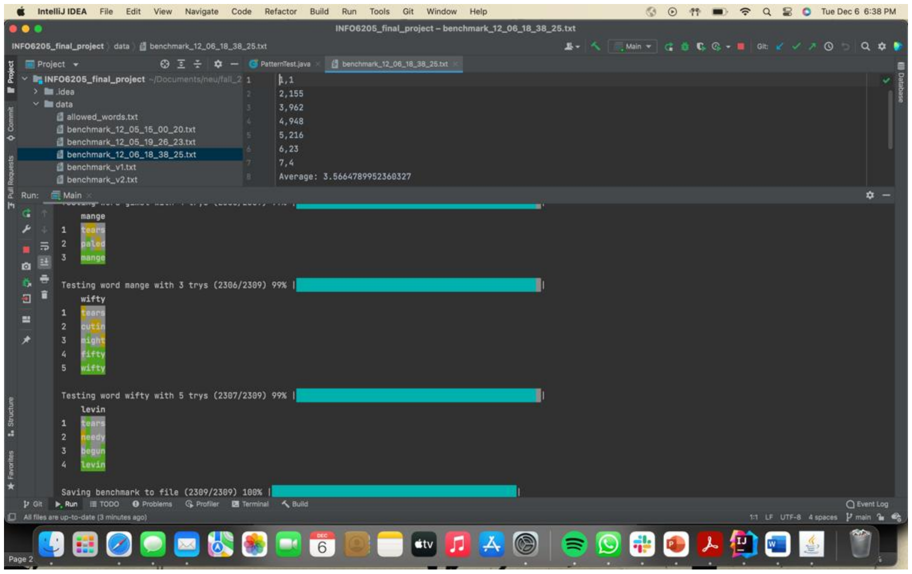


```
14855 possibilities, 13.86 bits   -5.69 bits
287 possibilities, 8.16 bits      -3.36 bits
28 possibilities, 4.81 bits       -1.22 bits
12 possibilities, 3.58 bits       -3.58 bits
1 possibilities, 0.00 bits        0.00 bits
1 possibilities, 0.00 bits
```

Top pick	E [info]	P
wharf	-0.000	0.001

3. Keep trying words to get all green pattern.

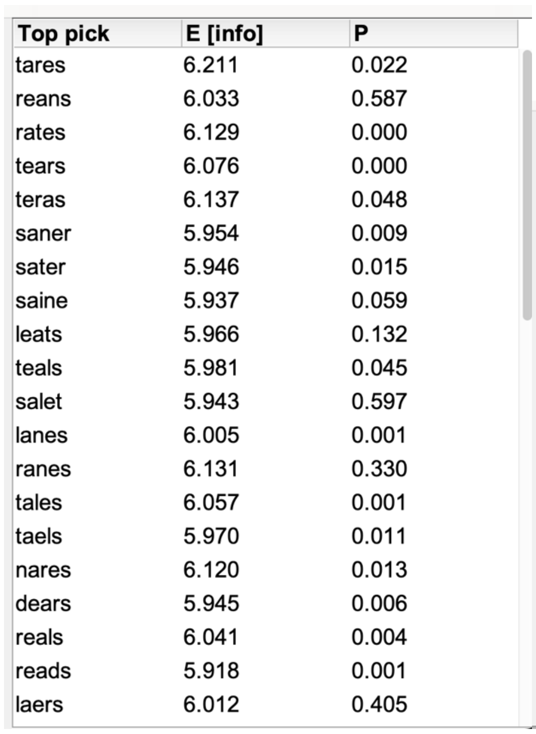
Benchmarking screen shots:


IntelliJ IDEA File Edit View Navigate Code Refactor Build Run Tools Git Window Help
INFO6205_final_project – benchmark_12_06_18_38_25.txt
INFO6205_final_project data benchmark_12_06_18_38_25.txt
Project
INFO6205_final_project ~/Documents/neu/fall_2
.idea
data
allowed_words.txt
benchmark_12_05_15_00_20.txt
benchmark_12_05_19_26_23.txt
benchmark_12_06_18_38_25.txt
benchmark_v1.txt
benchmark_v2.txt
PatternTest.java
Run: Main
Git Run TODO Problems Profiler Terminal Build
All files are up-to-date (3 minutes ago)

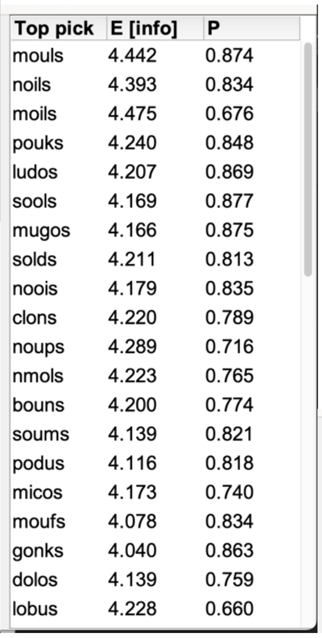
```
1,1
2,155
3,962
4,948
5,216
6,23
7,4
Average: 3.5664789952360327
```

```
    mange
1   tears
2   paled
3   mange

Testing word mange with 3 trys (2306/2309) 99% |
    wifty
1   tears
2   cutin
3   might
4   fifty
5   wifty

Testing word wifty with 5 trys (2307/2309) 99% |
    levin
1   tears
2   needy
3   begun
4   levin

Saving benchmark to file (2309/2309) 100% |
```

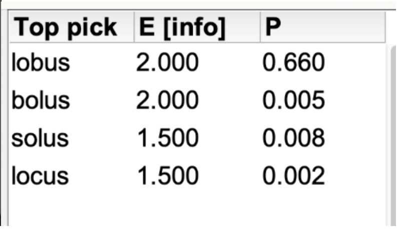
Entropy table screen shot:



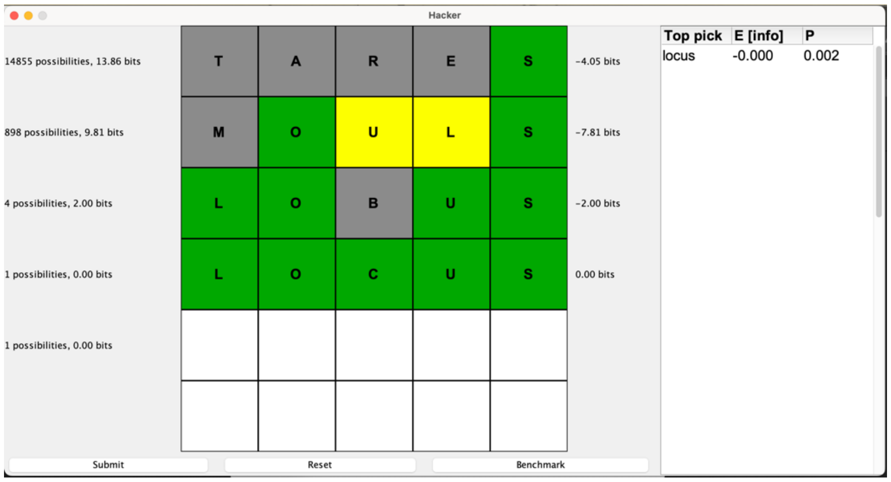
Top pick	E [info]	P
tares	6.211	0.022
reans	6.033	0.587
rates	6.129	0.000
tears	6.076	0.000
teras	6.137	0.048
saner	5.954	0.009
sater	5.946	0.015
saine	5.937	0.059
leats	5.966	0.132
teals	5.981	0.045
salet	5.943	0.597
lanes	6.005	0.001
ranes	6.131	0.330
tales	6.057	0.001
taels	5.970	0.011
nares	6.120	0.013
dears	5.945	0.006
reals	6.041	0.004
reads	5.918	0.001
laers	6.012	0.405



Top pick	E [info]	P
mouls	4.442	0.874
noils	4.393	0.834
moils	4.475	0.676
pouks	4.240	0.848
ludos	4.207	0.869
sools	4.169	0.877
mugos	4.166	0.875
solds	4.211	0.813
noois	4.179	0.835
clons	4.220	0.789
noups	4.289	0.716
nmols	4.223	0.765
bouns	4.200	0.774
soums	4.139	0.821
podus	4.116	0.818
micos	4.173	0.740
moufs	4.078	0.834
gonks	4.040	0.863
dolos	4.139	0.759
lobus	4.228	0.660



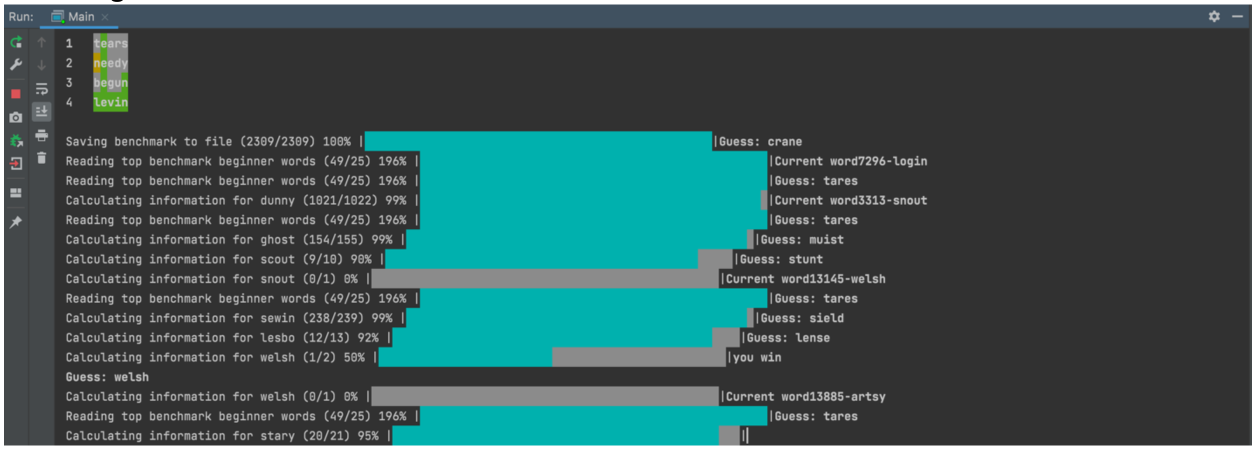
Top pick	E [info]	P
lobus	2.000	0.660
bolus	2.000	0.005
solus	1.500	0.008
locus	1.500	0.002



```
14855 possibilities, 13.86 bits    T A R E S    -4.05 bits
898 possibilities, 9.81 bits       M O U L S    -7.81 bits
4 possibilities, 2.00 bits         L O B U S    -2.00 bits
1 possibilities, 0.00 bits         L O C U S    0.00 bits
1 possibilities, 0.00 bits
```

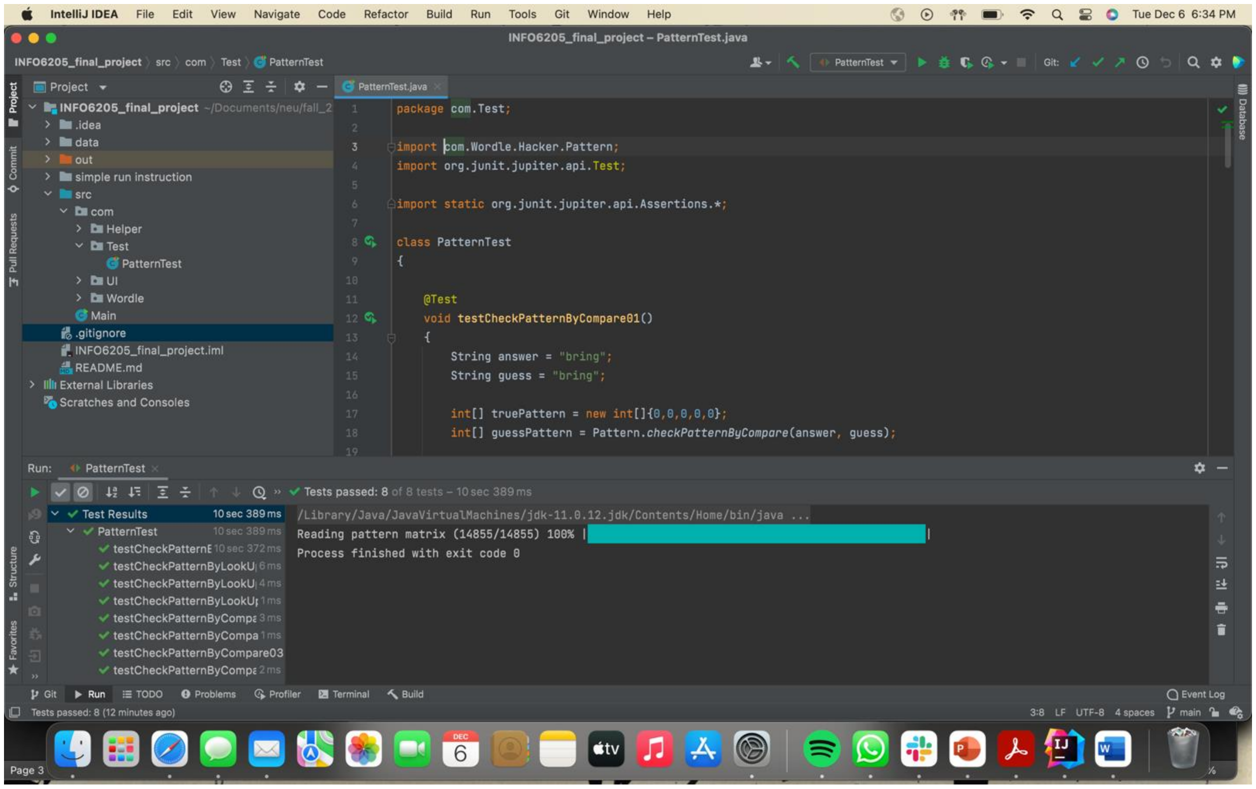
Top pick	E [info]	P
locus	-0.000	0.002

Move logs:



```
1   tears
2   needy
3   begun
4   levin

Saving benchmark to file (2309/2309) 100% |                    |Guess: crane
Reading top benchmark beginner words (49/25) 196% |            |Current word7296-login
Reading top benchmark beginner words (49/25) 196% |            |Guess: tares
Calculating information for dunny (1021/1022) 99% |            |Current word3313-snout
Reading top benchmark beginner words (49/25) 196% |            |Guess: tares
Calculating information for ghost (154/155) 99% |              |Guess: muist
Calculating information for scout (9/10) 90% |                 |Guess: stunt
Calculating information for snout (0/1) 0% |                   |Current word13145-welsh
Reading top benchmark beginner words (49/25) 196% |            |Guess: tares
Calculating information for sewin (238/239) 99% |              |Guess: sield
Calculating information for lesbo (12/13) 92% |                |Guess: lense
Calculating information for welsh (1/2) 50% |                  |you win
Guess: welsh
Calculating information for welsh (0/1) 0% |                   |Current word13885-artsy
Reading top benchmark beginner words (49/25) 196% |            |Guess: tares
Calculating information for stary (20/21) 95% |                ||
```

Unit test:



```
package com.Test;

import com.Wordle.Hacker.Pattern;
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

class PatternTest
{

    @Test
    void testCheckPatternByCompare01()
    {
        String answer = "bring";
        String guess = "bring";

        int[] truePattern = new int[]{0,0,0,0,0};
        int[] guessPattern = Pattern.checkPatternByCompare(answer, guess);
```

```
Tests passed: 8 of 8 tests – 10 sec 389 ms

Test Results                    10 sec 389 ms
  PatternTest                   10 sec 389 ms
    testCheckPatternE           10 sec 372 ms
    testCheckPatternByLookU     6 ms
    testCheckPatternByLookU     4 ms
    testCheckPatternByLookU     1 ms
    testCheckPatternByCompa     3 ms
    testCheckPatternByCompa     1 ms
    testCheckPatternByCompare03
    testCheckPatternByCompa     2 ms
```

```
/Library/Java/JavaVirtualMachines/jdk-11.0.12.jdk/Contents/Home/bin/java ...
Reading pattern matrix (14855/14855) 100% |                                        |
Process finished with exit code 0
```

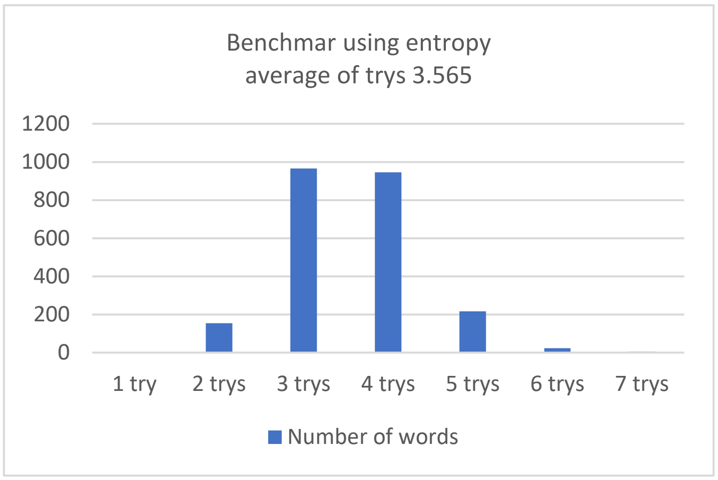
```
Tests passed: 8 (12 minutes ago)
```

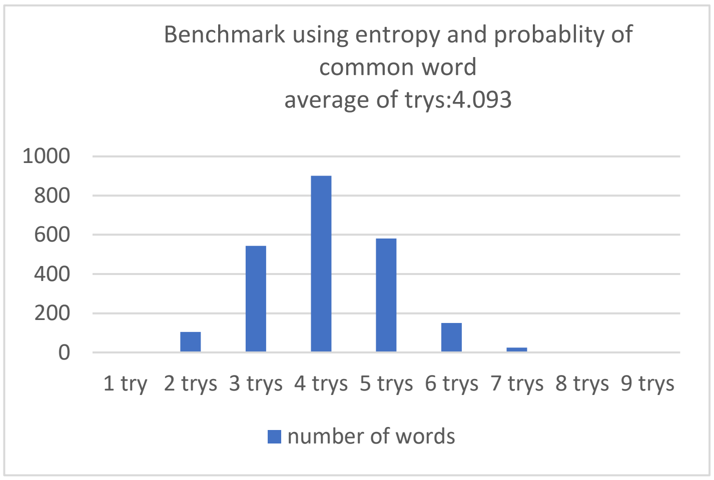
Analysis of Time Complexity for Calculating a word's Entropy:

We define fine that k is the number of letters in each word and n is the number of words in legitimate word list. There are 3 different statuses for each letter in a pattern. So totally there are 3^k different patterns. To comparing each 2 words, we need loop though all the letters for k^2 times to get their pattern. So, the time complexity of getting entropy of a word in a legitimate word list is $O(3^k * k^2 * n)$.

If we are not looping all the possible pattern, we just check the words and record the number of pattern occurrence. The time complexity will be the $O(k^2 * n)$ which Is the time to compare words times number of words.

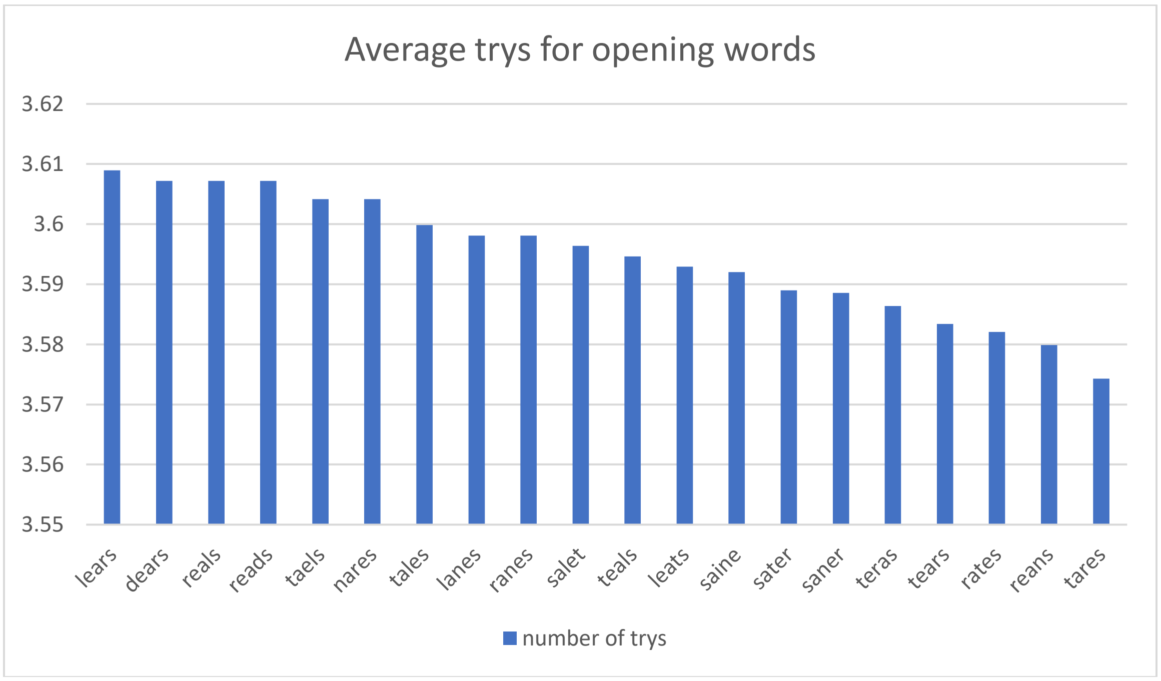
Conclusion:





Looping though all the words as the answer, we get the average number of tries used by the algorithm to give the answer.

In fact, after introducing the possibility of whether it is a commonly used word, the performance of the algorithm has decreased. This is different from the conclusion in the video. This may be because I used all possible 5 letters words list instead of the real wordle word list.



Our algorithm also calculates the maximum entropy in two steps to find the best starting word.

In the above figure, we can know that the opening word “tares” can make the operator find the answer in about 3.57 steps. This means, using our wordle hacker and start with “tares”, we can find the answer in 3 or 4 steps.

Reference:

- English word frequency data set:
  https://www.kaggle.com/datasets/rtatman/english-word-frequency?resource=download
- Wordle word list:
  https://github.com/tabatkins/wordle-list
- https://www.youtube.com/watch?v=v68zYyaEmEA%20%E2%80%A2%20
- https://www.youtube.com/watch?v=fRed0Xmc2Wg