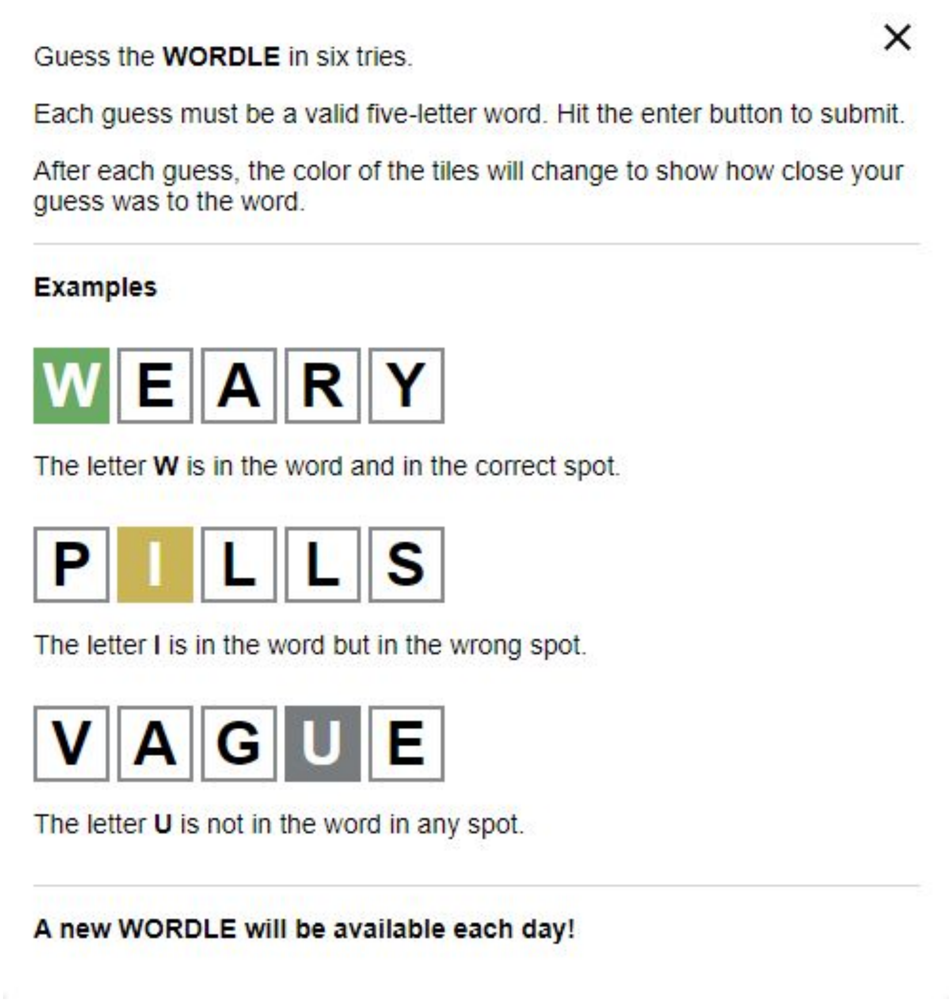
wordleBot

Anuhya Bhagavatula
Shrusti Ghela
Anagha Bandaru

Background

- Wordle is a web based word game that soared in popularity in 2022 and is currently owned and published by New York Times.
- The challenge of the game is to guess a five-letter word in six attempts.
- Each time a guess is made, the player is informed which of the chosen letters are in the target word.
    - The color green indicates that the letter is in the right position.
    - The color yellow indicates that the letter exists in the target word but is not in the right position.
    - The color black/grey indicates that letter does not exist in the target word.
- That’s it!



Guess the **WORDLE** in six tries.

Each guess must be a valid five-letter word. Hit the enter button to submit.

After each guess, the color of the tiles will change to show how close your guess was to the word.

Examples

W E A R Y

The letter **W** is in the word and in the correct spot.

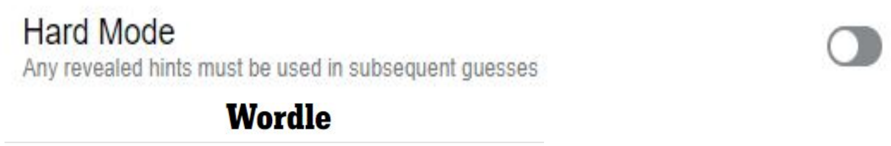
P I L L S

The letter **I** is in the word but in the wrong spot.

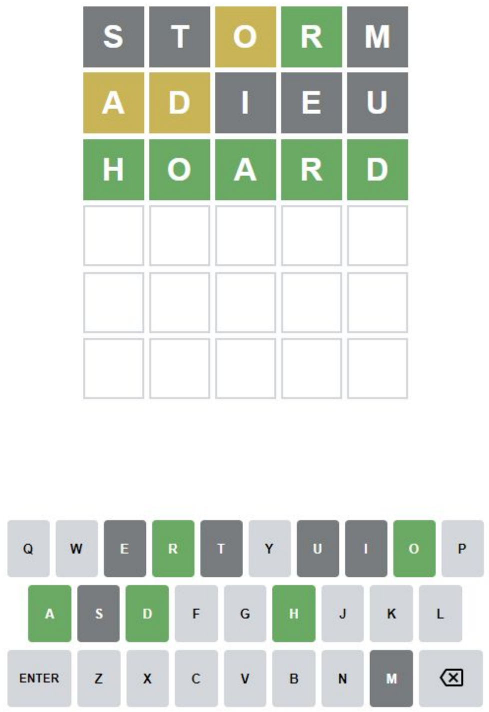
V A G U E

The letter **U** is not in the word in any spot.

A new WORDLE will be available each day!

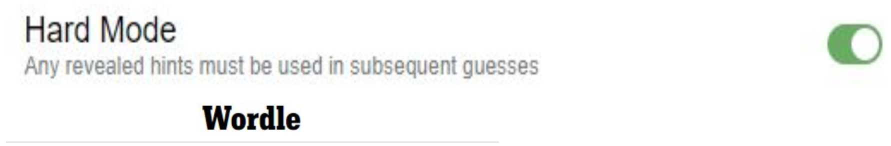

Hard Mode
Any revealed hints must be used in subsequent guesses

Wordle

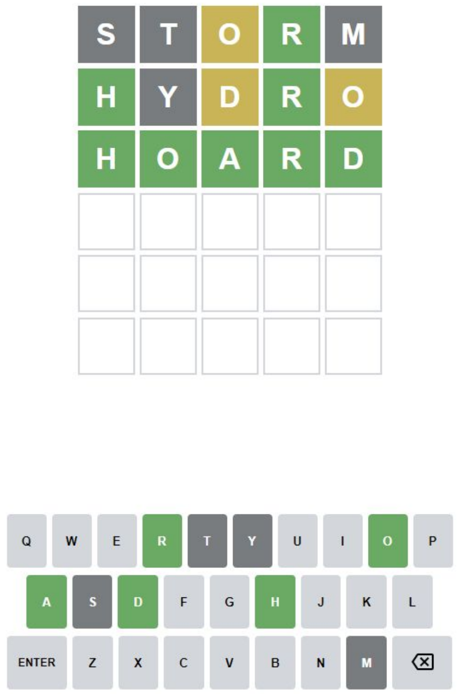


Wordle 261 3/6

⬜⬜🟨🟩⬜
🟨🟨⬜⬜⬜
🟩🟩🟩🟩🟩

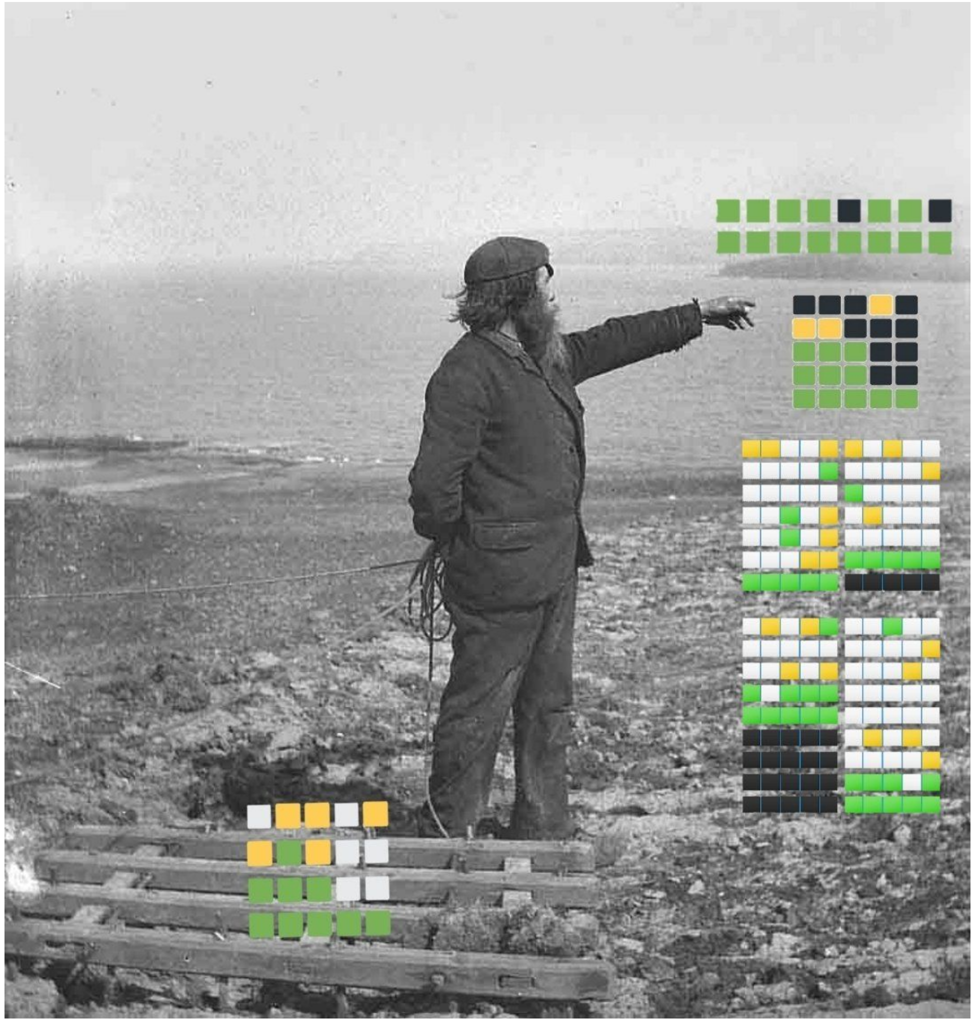

Hard Mode
Any revealed hints must be used in subsequent guesses

Wordle



Wordle 261 3/6*

⬜⬜🟨🟩⬜
🟩⬜🟨🟩🟨
🟩🟩🟩🟩🟩



Goal of wordleBot

In the project wordleBot, we create a Python package that can find an optimised solution for the given Wordle (optimistically in the first attempt) using the ⬛🟨🟩 tweet distribution.

This is done by simulations of hypothetical games and comparing the feedback received with data from Twitter by ranking a word based on cosine similarity.

The inspiration for this project is this excellent Ben Hammer's Kaggle project.

Link : https://github.com/anuhyabs/wordleBot

Datasets

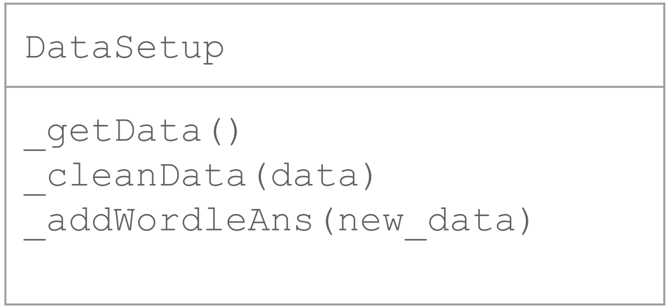
1. https://www.kaggle.com/rtatman/english-word-frequency

   This dataset contains the counts of the 333,333 most commonly-used single words on the English language web, as derived from the Google Web Trillion Word Corpus.

2. Twitter data collected using the Twitter API for a specific Wordle ID.

Design

We created a custom class - `DataSetup`, that is used to set up the data set of all possible answers and their frequency. In this class we have three functions.



```
DataSetup
_getData()
_cleanData(data)
_addWordleAns(new_data)
```

Design

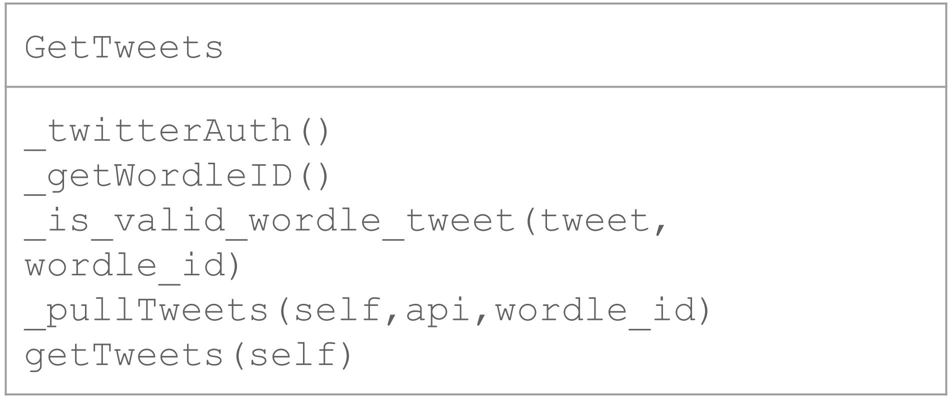
After the dataset was cleaned and ready to use, we created another custom class - `WordleSimulation`, that is used to simulate the wordle game for all the possible answers - the words in the dataset that we created using the `dataSetup` class.

WordleSimulation

`__init__(self)`
`_evaluate_guess_char(answer, guess, pos)`
`_evaluate_guess(self,answer, guess)`
`_simulate_wordle(self,answer,starting_weights)`
`_run_simulations(self,word, num_sims)`
`_getSimRes(self,sim_results)`
`_exportFiles(self,sim_vec_all,sim_vec_first,sim_vec_penultimate,sim_vec_ratio)`
`_invalidRes(self)`
`wordleSim(self)`

Design

We created another custom class - `GetTweets`, that is used to pulls a sample of tweets (5000) of a given Wordle id, filters those tweets that are valid, and adds it to the `tweets.csv` dataset



```
GetTweets

_twitterAuth()
_getWordleID()
_is_valid_wordle_tweet(tweet,
wordle_id)
_pullTweets(self,api,wordle_id)
getTweets(self)
```

Design

Finally, we created a custom class - `SolveWordle`, that is used to compute the distribution of Wordle results twitter for a particular id, compare this to the distributions we found from the wordleSimulatins class, rank the possible words based on the comparison of the distribution. The word the ranks the highest is our guess for solving the Wordle in the first try!

SolveWordle

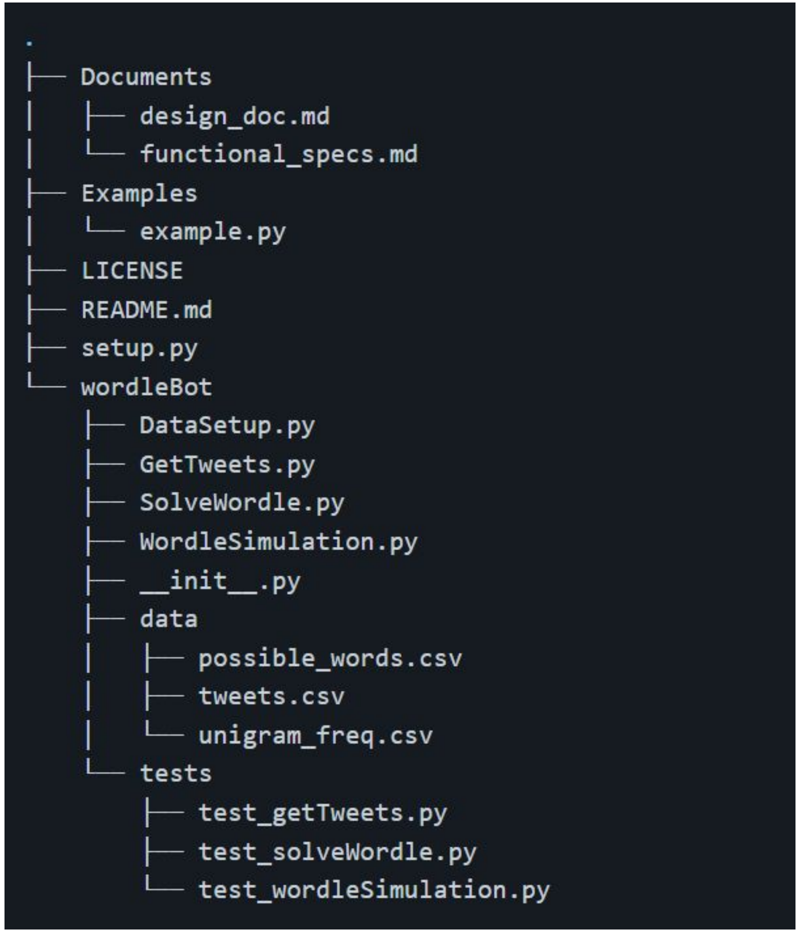
`__init__(self)`
`importData(self)`
`wordle_guesses(Tweet)`
`computeRank(self)`

Design Decisions

- Deep Modules
- Small Interface and larger implementations
- Separation of concerns
- Information hiding
- Adding docstrings to the code
- Code Extensibility
- Writing Unit tests for the code

Package Structure

- README.md file that gives an overview of the project.
- LICENSE file.
- setup.py file that initializes the project after it has been cloned.
- Documents folder that contains documentation of project.
- wordleBot folder that contains python modules and test files for the modules.
- Examples folder that contains examples of using the package



```
.
├── Documents
│   ├── design_doc.md
│   └── functional_specs.md
├── Examples
│   └── example.py
├── LICENSE
├── README.md
├── setup.py
└── wordleBot
    ├── DataSetup.py
    ├── GetTweets.py
    ├── SolveWordle.py
    ├── WordleSimulation.py
    ├── __init__.py
    ├── data
    │   ├── possible_words.csv
    │   ├── tweets.csv
    │   └── unigram_freq.csv
    └── tests
        ├── test_getTweets.py
        ├── test_solveWordle.py
        └── test_wordleSimulation.py
```

Further Works

- There is another efficient way of solving this problem - using Information Entropy. This could be integrated into the module.
- Our solution assumes that people play Wordle in the hard mode, where only words matching previously-given clues are considered for the next simulated guess. This could be addressed.

Thank you!