Rob Anderson
Jan 31, 2025

RL Model for Wordle

## Table of Contents:



# 1. Architecture

### 1.1 Overview

The Wordle Solver is structured as follows:

1. **Environment** (WordleEnvironment) – Responsible for providing the game state, rewards, and transitions in response to actions (guesses).

2. **Agent** (DQNAgent) – Learns an optimal policy through Deep Q-Learning, selecting actions (guesses) and updating a Q-Network.

3. **Neural Network** (QNetwork) – Approximates the Q-value function, which assigns expected cumulative rewards to (state, action) pairs.

4. **Replay Buffer** (ReplayBuffer) – Stores past experience tuples (s, a, r, s{\prime}, \text{done}) and enables mini-batch sampling for stable training.

5. **Training Loop** (train.py) – Orchestrates the episodes of interaction between Agent and Environment, and periodically evaluates performance.

6. **Main Entry Point** (main.py) – Provides a command-line interface for training or testing the agent, handling hyperparameters and I/O.

7. **Test/Game Play** (play_games.py) – Executes a series of test games, visualizes feedback, and provides statistics on performance

8. **Tests** (/tests) – a series of unit tests for this repository utilizing pytest

**Key Architecture Decisions:**

- State Representation:
  - Returns a *feedback matrix* of shape [5 x 26 x 3], capturing positions \times alphabet letters \times feedback categories (green, yellow, gray).
  - Maintains a boolean valid_mask indicating which words remain possible solutions given the feedback so far.

- Model Architecture
  - Approximate Q-values Q(s, a) for each possible guess, given the current environment state.
  - **Residual Layers**:
  - A residual connection around each hidden block (two layers) allows deeper networks to train more effectively by mitigating vanishing gradients.
  - **He Initialization**:
  - For ReLU activations, He (Kaiming) initialization ensures better performance in deep networks.
  - **Dropout**:
  - A small dropout (0.1) is introduced to help prevent overfitting.
  - **Input/Output**:
  - Input dimension (input_dim) is the flattened state (feedback matrix + valid mask + remaining guesses).
  - Output dimension (output_dim) is the size of the valid word list (each action corresponds to guessing a particular word).

- Reward Structure
  - Gives partial points for each green (2 points) and yellow letter (1 point).
  - Slight negative penalty (-0.1) for gray letters to encourage efficient guesses.
  - Bonus +10 plus an additional 2x remaining_guesses if the guess exactly matches the secret word.

- Training:
  - **Number of Episodes**: 1,000 (adjustable).
  - **Max Guesses per Episode**: 6 (Wordle standard).
  - **Replay Batch Size**: 64.
  - **Target Network Update Frequency**: 100 steps.
  - **Evaluation Frequency**: 100 episodes.
  - **Evaluation Episodes**: 100 (test puzzles each evaluation)

## 2. Performance

**Train Time:**
500 words: 2 min

**Test Performance:**
500 different words

- 20/20 correct
- Average Guesses: 3.35

Most Common First Word Guess: *FUTON*

## 3. Project Work Details

**Time Tracking:**
Total time 11 hours:

4. **1 hr:** Initial Research 3:15-4:15pm Saturday Jan 18, 2025
    a. Used GPT o1 and Gemini 2.0 and 1.5 Deep Research to get an understanding of the space and techniques
5. **1.5 hour** State Space Research 10:30am-1:00pm Sunday Jan 19, 2025
    a. Used Claude Opus and compared to GPT o1 and Gemini and Decided on State Space (5x26 binary matrix) and algorithm Q learning
6. **1 hour:** 1:30pm-2:30  Sunday Jan 19, 2025: implementation through windsurf
    a. I had a model training within 15 min – have no idea what it's doing, but it's working
    b. Code is in Windsurf– but need to dive deeper into it and understand it better.
7. **2 Hours** 4:00pm – 6:00pm Sunday Jan 19, 2025: understand code and make improvements
    a. It works!
8. Got it working within **5.5 hours…**
9. **1 hour:** 12-1pm Monday Jan 20, 2025: got DQN version going through o1 and windsurf
10. **2 hours:** 3-5pm Monday Jan 20, 2025: work on visualization tool to compare the two systems.
11. **1 hour:** 9-10pm Monday Jan 20, 2025: documentation of architectures
12. **2 hours** 8-10pm Sunday Jan 26, 2025: Work on multi-agent - could not get it

## 13. Research Notes

# Approach

## Assumptions:

1) Compute Budget
2) Given list of words possible
3) Standard Wordle Rules

## Architecture:

- In RL, you always start by specifying your Markov Decision Process (MDP):
  - The *state space* (what you observe and how it's encoded),
  - The *action space* (what moves the agent can make),
  - The *transition dynamics* and reward function.

1) First Decision - How to Represent the State
   a) Option 1:
      i) As an array of strings of guesses and feedback
         (1) A combination of three sets, (1) a 5-element list representing the known letter positions (e.g., [_, _, A, _, _] for a word where the third letter is A), (2) a set of letters known to be in the word but without known position (yellow letters), (3) a set of letters known not to be in the word (gray letters), and (4) a list of past guesses with feedback.
      ii) Pro's
         (1)
      iii) Con's
   **b) Option 2:**
      **i) As a 5x26 binary matrix, list of remaining possible guesses, how many remaining turns**
         (1) Remaining words are represented as a list of strings
         (2) or
         (3) Words are represented as binary bag of words for generalization
         (4) Also with letter frequency vector
      ii) Additional Compressions to try out:
      iii) Pro's
         (1)
      iv) Con's

            (1)
    c) Go with Option 2
2) 2nd Decision – what algorithm to use:
    a) Q Learning
        i) Adfa
        ii) Reward Structure
            (1) Use reward shaping or a curriculum. For example, give partial credit for each correct letter identified, **or for reducing the candidate set**.
            (2) Or keep it simple with a high reward for success, a small penalty for failing, and rely on enough training episodes for the model to converge.
        iii) Pro's
            (1) D
        iv) Con's
            (1)
    b) DQN
        i) If the state space becomes too large for Q-learning, we can consider DQN, which uses a neural network to approximate the Q-function.
        ii) Reward Structure
        iii) Pro's
            (1) Da
        iv) Con's
            (1) daf
    c) Policy Gradient
        i) Adaf
        ii) Reward Structure
        iii) Pro's
            (1) Da
        iv) Con's
            (1) Daf
    d) LSTM
        i) Possibly an LSTM or Transformer-based approach to manage the combinatorial nature of letters and positions.
    e) Hybrid Approach
        i) Use heuristics for early guesses (like words that maximize letter coverage), and only switch to RL-based selection once the solution space is narrowed.
            (1) Minimize the search space for first two guesses
        ii) Pro's
            (1) Cad
        iii) Con's
            (1) Adfa
    f) Rather than large transformers, start with a modest feed-forward network or small RNN that takes your observation as input and outputs action probabilities.
    g) MCTS
    h) Training Exploration

        i) **Epsilon-greedy exploration:** With a probability of epsilon, the agent chooses a random action (word) from the action space. Otherwise, it selects the action with the highest estimated value according to its Q-function.
        ii) **Softmax exploration:** The agent assigns a probability to each action based on its estimated value and then samples an action from this probability distribution. This allows for a more nuanced exploration strategy, where actions with higher estimated values are more likely to be chosen but not guaranteed.

3) Ok, now we have a strategy in place - implementation time
    a) Start with a small sample to prove out approach
        i) 200 words
    b) Dev Environment
        i) Windsurf with Claude
    c) Build out infrastructure
        i) Game Play
            (1) Python
                (a) Fast enough
                (b) Easy to iterate
        ii) Unit Tests
        iii) Training Infrastructure
            (1) PyTorch
4) Test Results
    a) Dafda
5) What did we learn?
    a) How to update the strategy
    b) Implement with 800 words
        i) Keep Q algorithm as a base
        ii) Start over
    c) Test on 200 words not in training set
6) Test Results
    a) Dafa
7) Any more updates?
8) Challenge Mode
    a) Multi-Agent RL

```
+-----------------+       +-----------------+       +-----------------+
|   Start Game    |------>|  Initialize     |------>| Select Action   |
|                 |       |  State (empty) |       | (Epsilon-Greedy)|
+-----------------+       +-----------------+       +-----------------+
                                    ^                      |
                                    |                      |
                                    |                      v
+-----------------+       +-----------------+       +-----------------+
| Update Q-Table  |<------| Receive Reward  |<------| Take Action     |
| (based on       |       | & Next State   |       | (Guess Word)    |
|  reward)        |       +-----------------+       +-----------------+
+-----------------+                |                      |
      ^                      |                      v
      |                      |          +-----------------+
      +----------------------+          | Wordle          |
                                        | Environment     |
                                        | (Feedback)     |
                                        +-----------------+
                                                |
                                                v
                                        +-----------------+
                                        | Game Over?      |
                                        +-----------------+
                                                |
                                                v
                                          Yes / No
                                        +-----------------+
                                        | End Game        |
                                        +-----------------+
```