ML Hackathon

GROUP 6:

AKASH MADISETTY - PES1UG23AM035

AMIT PRAKASH - PES1UG23AM042

ANKUR GUPTA - PES1UG23AM054

AHANA SHETTY - PES1UG23AM030

Key Observations

Challenging Parts:

Building a reliable Hangman RL environment that works both for Gym-style agent interaction and for various forms of letter-guessing policies was complex, especially ensuring API consistency for reset, step, and state features.

Training and integrating an HMM character-level model required careful preprocessing and diagnostic plotting, and debugging shape mismatches or missing HMM parameters demanded substantial effort.

The reward design's impact on RL agent learning performance (balancing correct guesses, wrong guesses, repeat penalties, etc.) was critical and took time to tune.

Insights Gained:

Incorporating hand-crafted, interpretable features—like the HMM oracle's letter probabilities—significantly improves agent sample efficiency compared to using only one-hot or purely masked word encodings.

Even with a strong signal from the HMM, exploration remains vital for long-tail words and rare patterns, demonstrating the enduring importance of "exploration vs. exploitation" in RL.

Strategies

- ### HMM Design Choices:

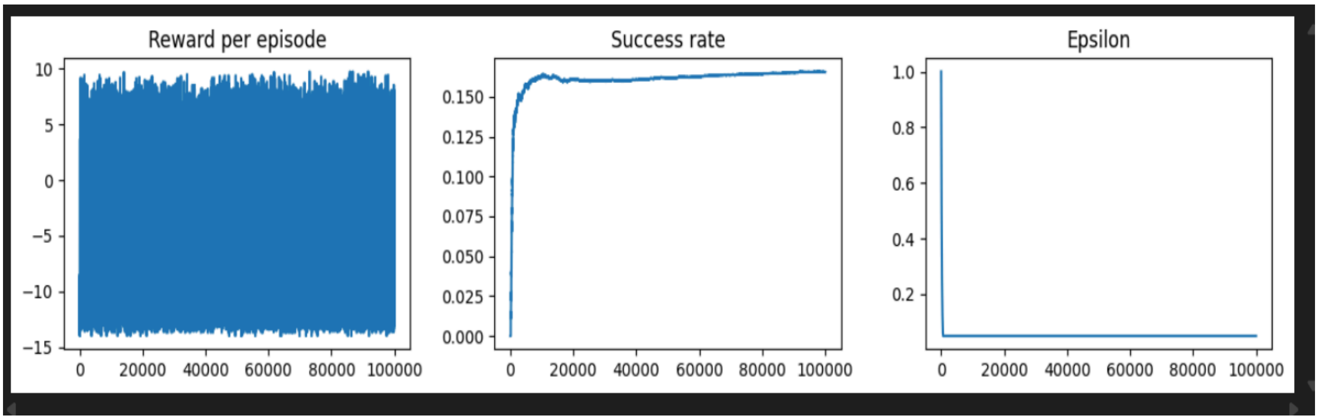
We trained a Multinomial HMM on large word lists, using letter-level emissions and position information.

If the HMM was unavailable or incorrectly shaped, we fell back to frequency-based or uniform oracles, but the intended method aggregates marginal emission probabilities over possible hidden states for each blank letter.

- ### RL State and Reward Design:

State: Combined the masked word, guessed letters, remaining lives, and the HMM-provided letter probability distribution, offering the RL agent all observable game-relevant information in compact form.

Reward: Positive reward for correct guesses (+2), bonus for completing the word (+10), penalty for wrong guesses (-1), repeat guesses (-2), with strong negative reward for losing (-10). This reward balances winning, guessing efficiently, and exploring new actions.

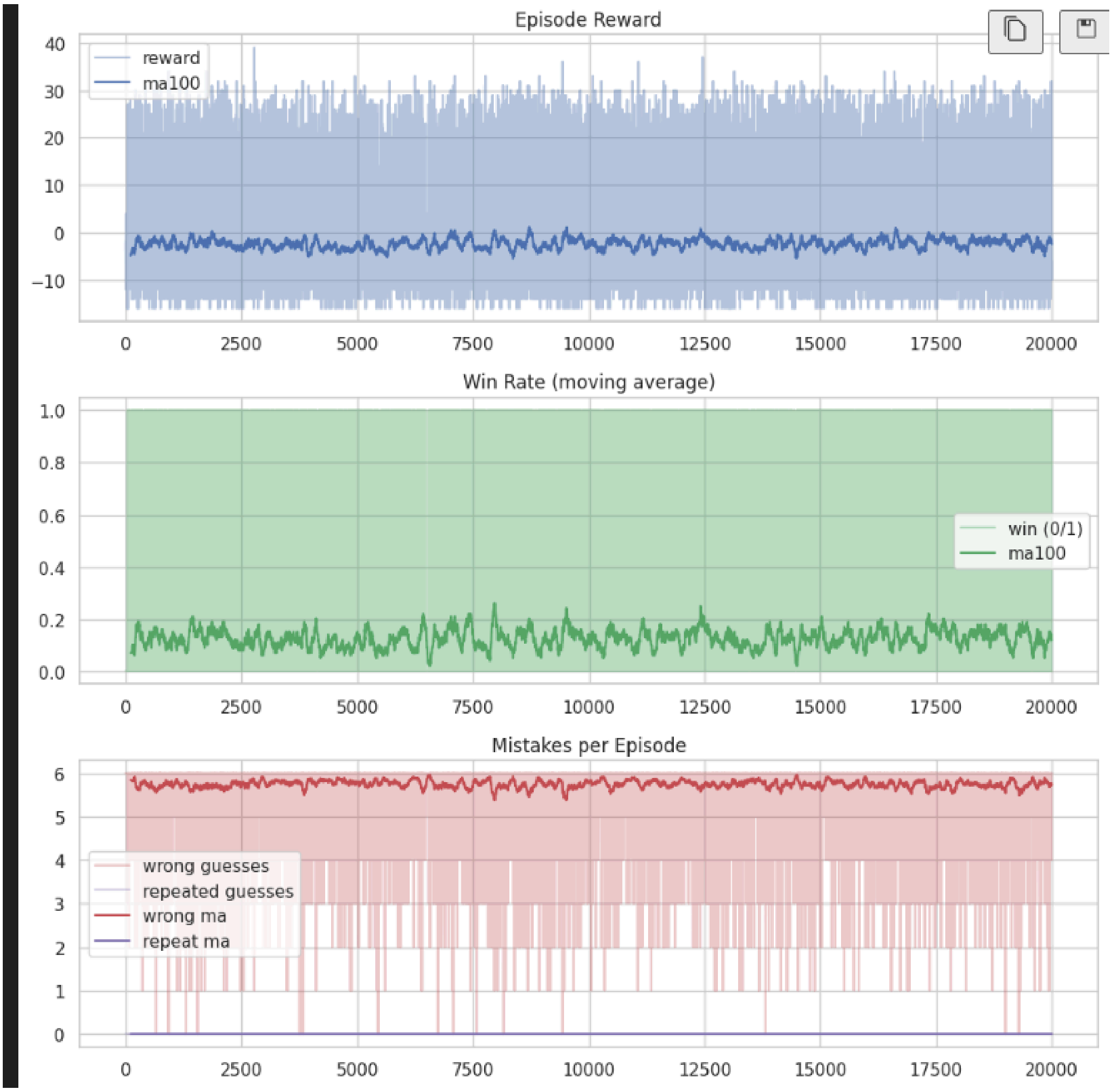


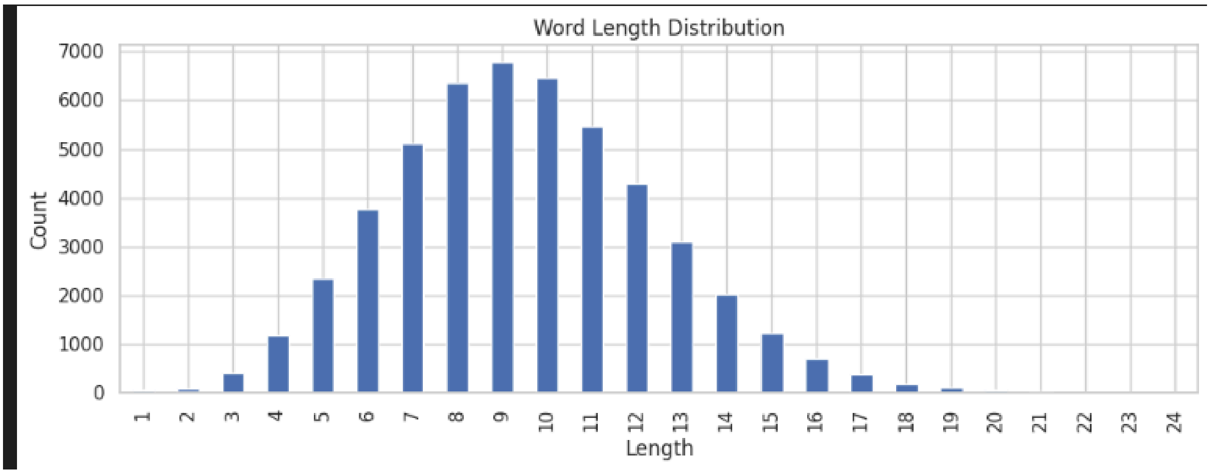
Exploration vs. Exploitation

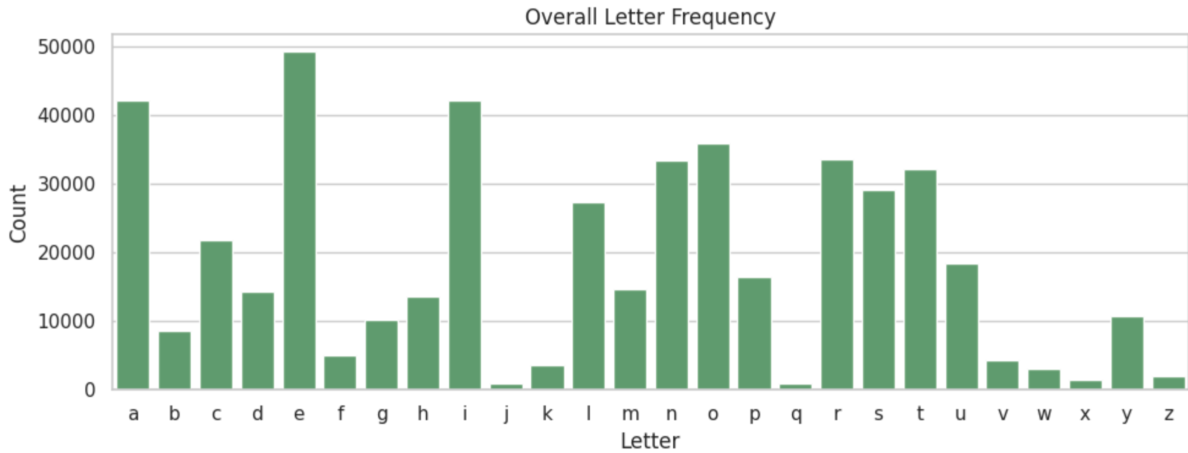
- Your Approach:

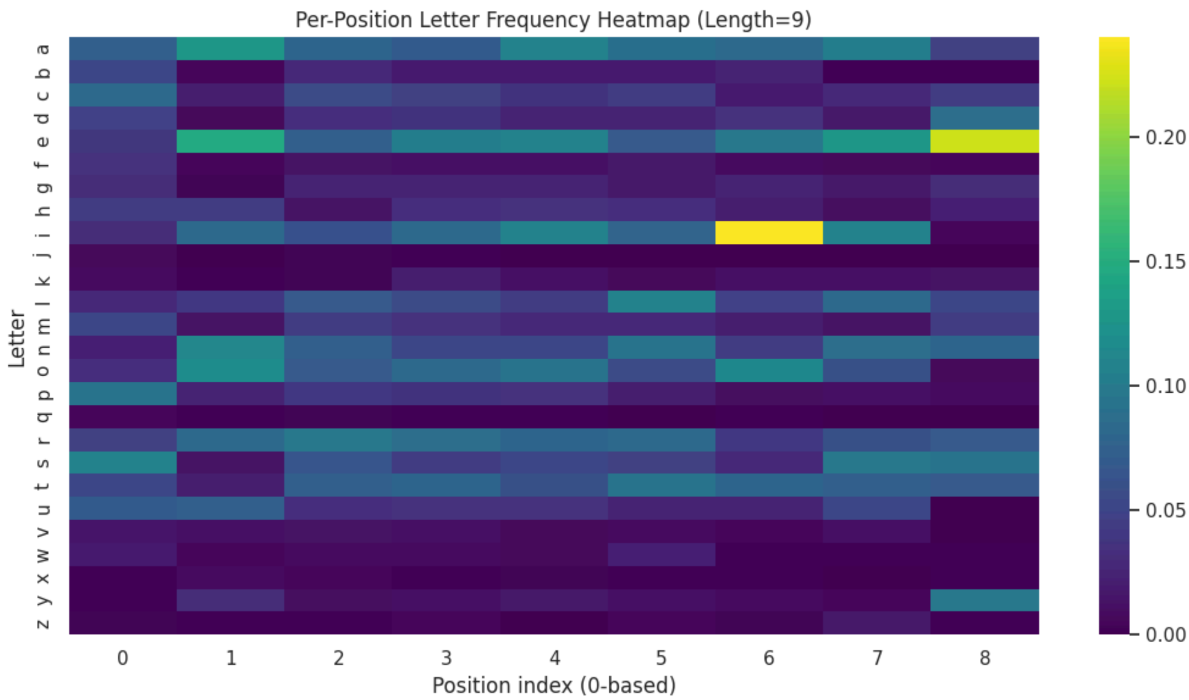
Used epsilon-greedy action selection: with probability ε, the agent "explores" by sampling guesses based on the HMM oracle's letter probabilities; with probability 1-ε, the agent exploits by choosing the action with the highest Q-value ("best guess") for the current state.

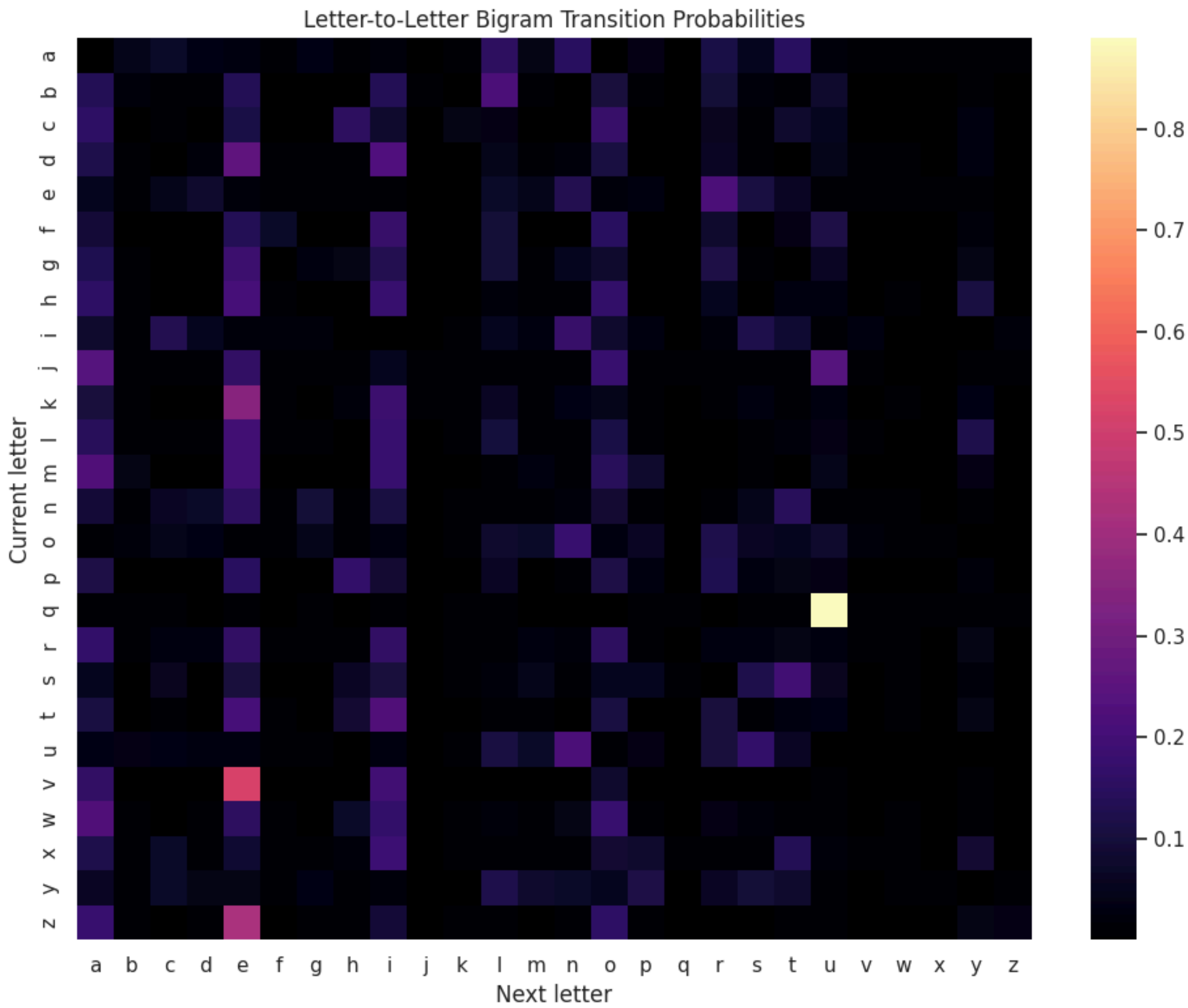
ε was decayed from 1.0 to 0.05 to transition from exploration to exploitation over training, striking a balance between discovering new strategies and leveraging those that maximize cumulative reward.



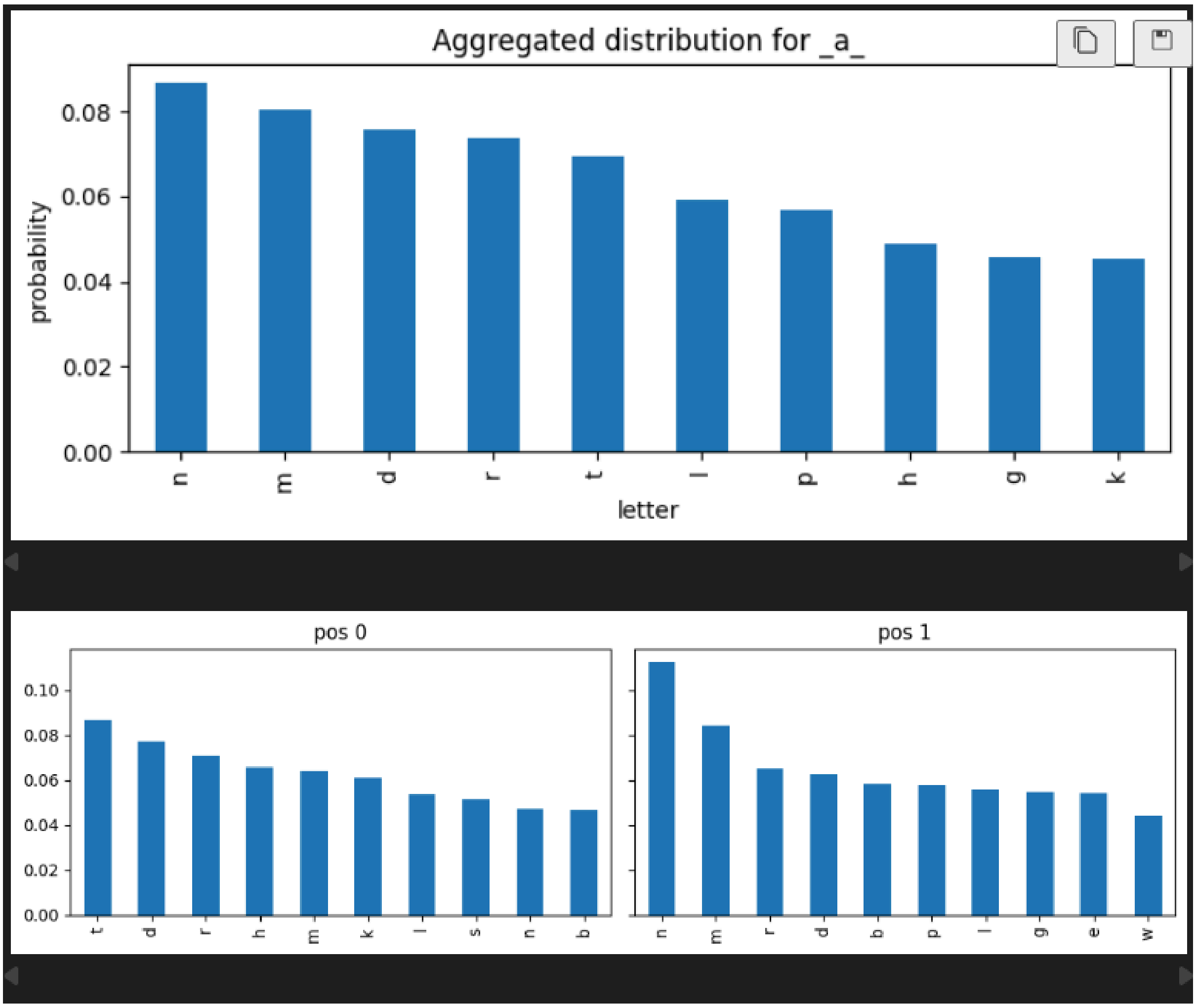

Word Length Distribution
Count
Length
0
1000
2000
3000
4000
5000
6000
7000
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24


Overall Letter Frequency
Count
Letter
0
10000
20000
30000
40000
50000
a b c d e f g h i j k l m n o p q r s t u v w x y z


Per-Position Letter Frequency Heatmap (Length=9)
Letter
a b c d e f g h i j k l m n o p q r s t u v w x y z
Position index (0-based)
0 1 2 3 4 5 6 7 8
0.00
0.05
0.10
0.15
0.20



Letter-to-Letter Bigram Transition Probabilities
Current letter
a b c d e f g h i j k l m n o p q r s t u v w x y z
Next letter
0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8


Aggregated distribution for _a_
probability
letter
0.08
0.06
0.04
0.02
0.00
n m d r t l p h g k
pos 0
pos 1
0.10
0.08
0.06
0.04
0.02
0.00
t d r h m k l s n b
n m r d b p l g e w


Episode vs Win rate — compare window sizes
Win rate (moving average)
Episode
1.0
0.8
0.6
0.4
0.2
0.0
0
2000
4000
6000
8000
10000
12000
14000
Window
MA w=1
MA w=10
MA w=50
MA w=100
MA w=200

Result:

```
BEST RUN:
params = {'EPISODES': 15000, 'ALPHA': 0.05, 'EPS_DECAY': 0.999, 'EPSILON': 0.3, 'seed': 123}
success_rate = 0.233
final_score = -54284.0
```

Future Improvements With Another Week:

Train a deeper or more accurate HMM (using a larger corpus, tuning state count, etc.)

Incorporate n-gram or neural models for richer prior guesses.

Try more sophisticated RL algorithms (e.g., DQN, Actor-Critic) and richer features.

Implement robust analytics: Analyze game failures, top-k accuracy by word length/frequency, and confusion matrices for agent errors

Collaborate to tune reward signals specifically for rare words and edge cases.

Extend evaluation to include human-competitive benchmarks and adversarial test sets.