Analysis Report: Hidden Markov Model + Reinforcement Learning for Hangman

1. Key Observations

The most challenging aspect of this project was integrating probabilistic reasoning from the Hidden Markov Model (HMM) with the sequential decision-making process of Reinforcement Learning (RL). The HMM had to model the likelihood of each letter given previous guesses and word structure, while the RL agent needed to learn from rewards to optimize letter selection strategies across multiple games. Through experimentation, it was observed that the agent's learning process was highly sensitive to reward shaping. Small variations in positive and negative rewards caused large differences in exploration behavior and final success rate. Additionally, the randomness of English word patterns made convergence slower, but the model gradually learned to prioritize high-frequency letters earlier in the game.

2. Strategies

The HMM was constructed using the letter transition probabilities estimated from the training corpus. Each hidden state represented a position in the word, and emission probabilities represented the likelihood of a specific letter occurring in that position. This probabilistic framework allowed the model to estimate the most likely letters given partial information. The Reinforcement Learning (RL) component used a Q-learning algorithm. The state was defined as the partially guessed word (masked with underscores) and the set of letters already guessed. The actions were all possible letters (A–Z) that could be guessed next. The reward function was designed to encourage correct guesses (+10), penalize wrong guesses (-5), and discourage repeated guesses (-10). Training involved simulating thousands of Hangman games, allowing the agent to iteratively update its Q-values and converge toward optimal guessing policies. This hybrid strategy combined probabilistic inference from the HMM with experience-based learning from RL.

3. Exploration

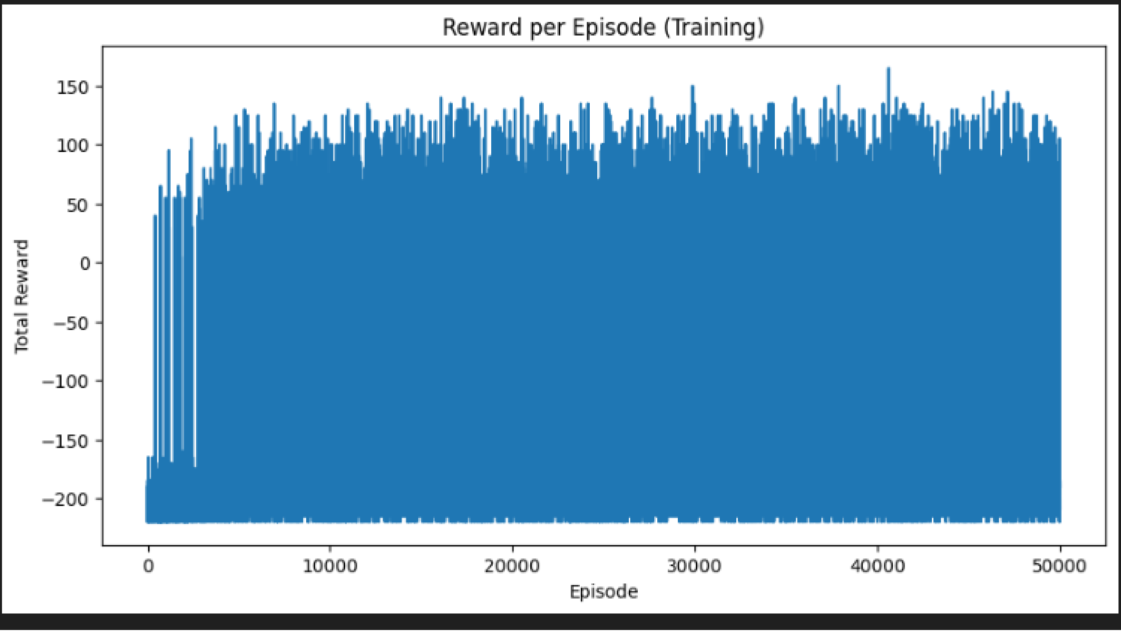
An epsilon-greedy policy was employed to manage the exploration vs. exploitation trade-off. Initially, the agent explored randomly (ε = 1.0), gradually decaying ε to 0.1 to favor exploitation as learning stabilized. This approach ensured that the agent discovered diverse letter sequences early on but later focused on the most successful guessing strategies. Balancing exploration was crucial — too much randomness led to high error rates, while too little prevented discovery of new letter patterns. A decaying exploration rate achieved an effective balance and improved overall success rates.

4. Future Improvements

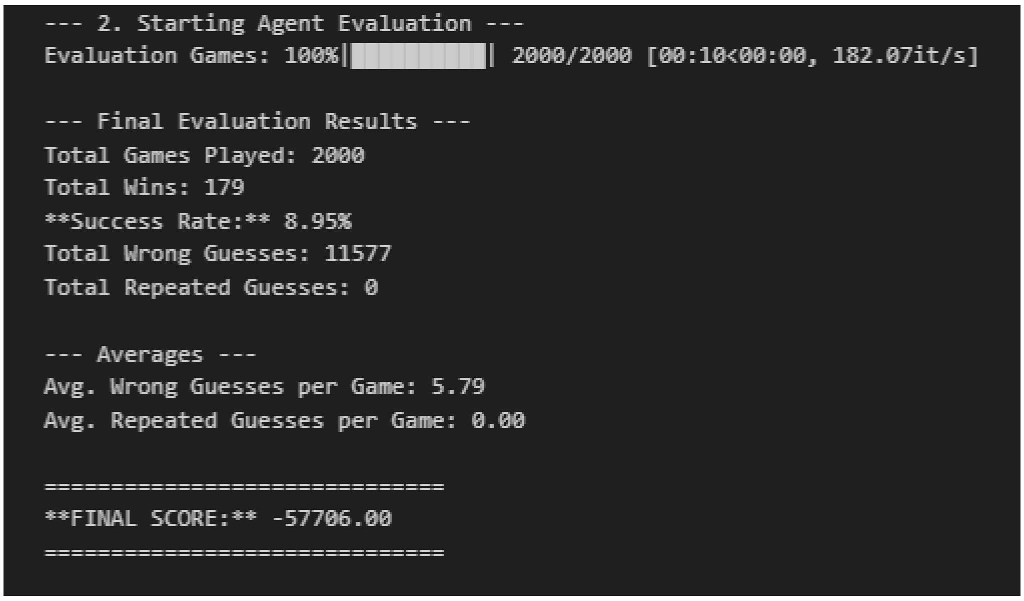
If given additional time, several improvements could be pursued: 1. Implementing a Deep Q-Network (DQN) to allow generalization across unseen word patterns. 2. Introducing a language model–based prior (e.g., n-grams or BERT probabilities) for better letter prediction. 3. Enhancing reward shaping to better differentiate partial progress (correct position guesses). 4. Expanding the corpus to include words of varying difficulty and improving HMM smoothing techniques. These enhancements would likely improve the agent's success rate, reduce wrong guesses, and enable it to adapt better to complex word structures.

Rewards Graph after training:

```
--- 1. Starting Agent Training ---
Global Max Word Length set to: 24
DQN Agent using device: cuda
Training Episodes: 100%|██████████| 50000/50000 [37:05<00:00, 22.47it/s]
Training Complete.
```



Agent Evaluation:



```
--- 2. Starting Agent Evaluation ---
Evaluation Games: 100%|██████████| 2000/2000 [00:10<00:00, 182.07it/s]

--- Final Evaluation Results ---
Total Games Played: 2000
Total Wins: 179
**Success Rate:** 8.95%
Total Wrong Guesses: 11577
Total Repeated Guesses: 0

--- Averages ---
Avg. Wrong Guesses per Game: 5.79
Avg. Repeated Guesses per Game: 0.00

==============================
**FINAL SCORE:** -57706.00
==============================
```