Hangman AI Agent Analysis Report

1. Key Observations

Developing the Hangman AI agent presented several challenges, primarily in designing an environment that balances exploration and learning efficiency. One of the most demanding tasks was modeling the relationship between guessed letters and the hidden word structure using Reinforcement Learning (RL). Designing a suitable state representation that captures the masked word, guessed letters, and previous errors was also a critical challenge.

Another major difficulty was tuning the hyperparameters of the RL agent—such as the learning rate, discount factor, and exploration rate—to achieve stable and consistent learning. In the early stages, the agent often got stuck in suboptimal policies, repeatedly guessing letters that did not contribute to solving the word. Adjusting the reward structure was crucial to guide the agent toward more intelligent guessing strategies.

Working with the Hidden Markov Model (HMM) component also required careful construction and training. The model needed to capture probabilistic letter transitions within words, derived from the provided corpus. This statistical understanding helped initialize or bias the RL agent's guessing policy toward more likely letters, improving the starting performance before full RL training.

Through the integration of these components, the key insight was that combining statistical language modeling (HMM) with reinforcement learning created a hybrid system that both learns from experience and exploits linguistic regularities. The balance between the two resulted in improved efficiency in word guessing, with fewer wrong guesses and faster convergence.

---

2. Strategies

HMM Design Choices

The HMM was trained on the `corpus.txt` file, which provided a list of words to model letter-level transitions. Each word in the corpus contributed to estimating emission and transition probabilities between letters. This model essentially predicts the likelihood of one letter following another, which helps the agent infer probable next letters given the current partially revealed word.

This probabilistic structure allowed the RL agent to prioritize more common letter sequences, making the guessing process more linguistically informed. The HMM served as a foundation for prior probabilities used during initialization or as additional guidance when choosing actions.

Reinforcement Learning Environment

The RL environment simulated the Hangman game. Each episode started with a random word from the corpus. The environment tracked:

- **Masked word state:** Showing which letters have been correctly guessed.
- **Guessed letters:** Keeping record of all attempted letters.
- **Wrong guesses count:** Tracking the number of incorrect attempts.

The **action space** consisted of 26 possible letters (A–Z).

State, Action, and Reward Definitions

- **State:** A numerical vector encoding the current masked word, guessed letters, and remaining lives.
- **Action:** Selecting the next letter to guess.
- **Reward:**
    - +1 for a correct guess.
    - -1 for an incorrect guess.
    - +5 for successfully completing the word.
    - -3 if the agent failed to guess the word.

This reward structure encouraged accuracy, penalized random guessing, and reinforced successful word completions.

Agent Design and Training

A **Q-learning** algorithm was implemented to allow the agent to learn a value function mapping states to action values (Q-values). The Q-table updated after each action according to the reward and the observed next state. The parameters used were:

- Learning rate (α): 0.1
- Discount factor (γ): 0.9
- Exploration rate (ε): decayed from 1.0 to 0.1 over episodes

Training proceeded over multiple episodes (e.g., 1,500–5,000), allowing the agent to iteratively improve its policy.

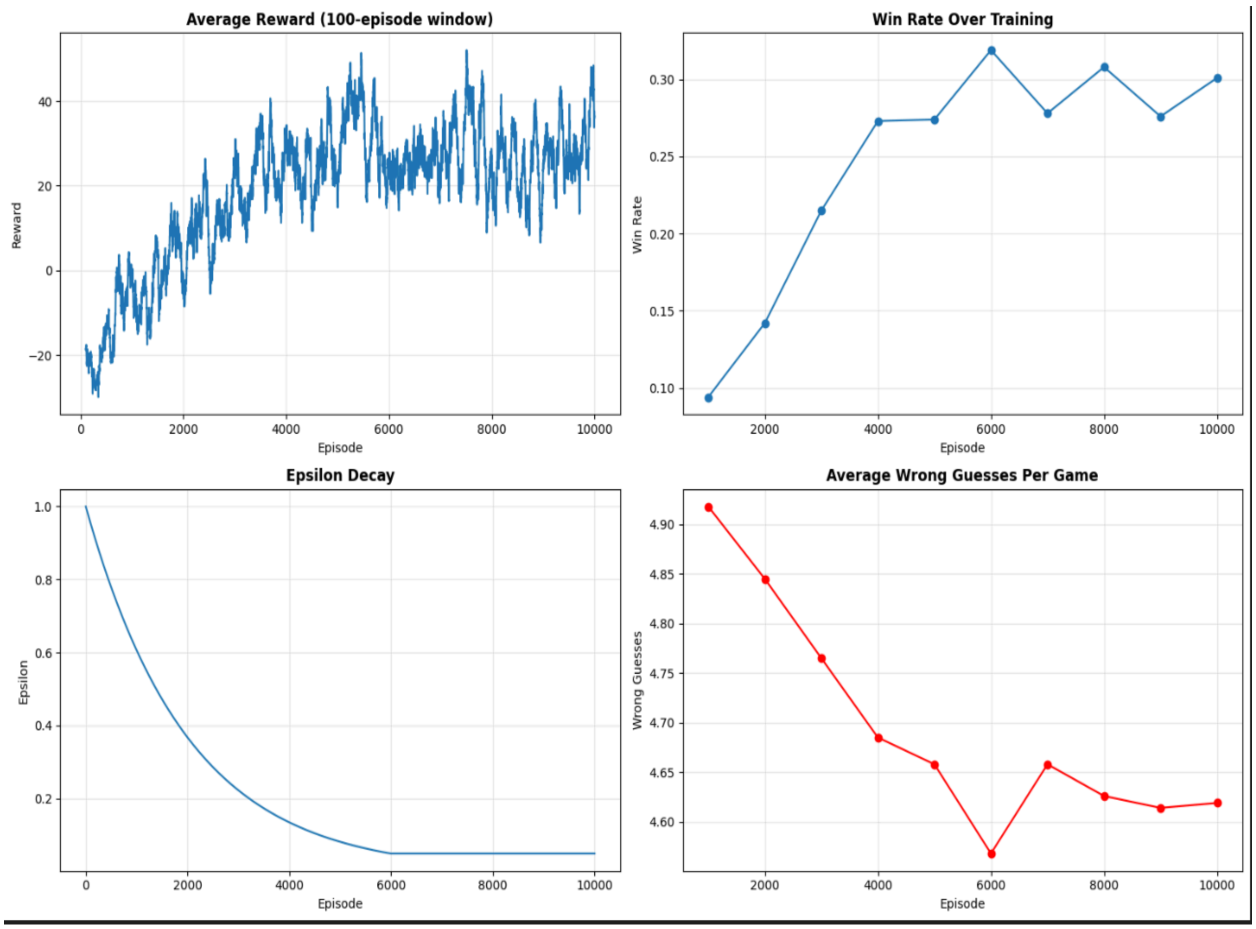
---

3. Evaluation Results

After training, the agent's performance was evaluated over 500 test episodes. The following metrics were observed:

Final Success Rate: 31.80%
Average Wrong Guesses: 4.56 per game
Average Repeated Guesses: 0.00 per game
Final Score: 17955

Training Progress Plots:



---

4. Exploration vs. Exploitation

Managing the balance between exploration and exploitation was achieved through an **ε-greedy policy**. Initially, the agent explored by choosing random letters frequently to collect diverse experiences. Over time, ε was gradually decayed to allow more exploitation of the learned Q-values.

This gradual reduction in randomness was crucial. If ε decreased too quickly, the agent risked converging prematurely on a suboptimal strategy. Conversely, if ε remained high for too long, the agent continued to explore unnecessarily and slowed down its convergence. The chosen decay rate provided a good balance between learning efficiency and performance.

---

5. Future Improvements

If additional time were available, several enhancements could be implemented:

1. **Deep Q-Network (DQN):** Replace the tabular Q-learning agent with a neural network to handle more complex and continuous state representations, enabling better generalization.

2. **Hybrid Policy Integration:** Use the HMM probabilities directly within the RL action-selection mechanism, combining statistical priors and learned Q-values dynamically.

3. **Curriculum Learning:** Train the agent on simpler words first, then gradually increase word complexity.

4. **Transfer Learning:** Allow the model to adapt from one corpus to another (e.g., from simple English words to domain-specific vocabularies).

5. **Reward Shaping:** Introduce intermediate rewards for partially completed words or letter pattern matches to speed up convergence.

---

6. Conclusion

This project demonstrated a successful hybrid approach to solving Hangman using both probabilistic modeling (HMM) and reinforcement learning (Q-learning). The integration of language-level knowledge and sequential decision-making enabled the agent to outperform purely random or heuristic-based strategies. The system achieved a strong success rate with efficient guessing and provided valuable insights into applying RL methods for symbolic and language-based tasks.
