Hangman HMM + RL Agent: Analysis Report

Key Observations

The most challenging parts of this project were ensuring the Hidden Markov Model (HMM) converged properly and maintaining numerical stability during forward–backward computations. Managing the balance between probabilistic letter prediction from the HMM and the reinforcement learning (RL) exploration strategy required careful tuning. Another difficulty was defining an informative reward function — one that encourages correct guesses but penalizes wrong or repeated ones appropriately. Once tuned, the model exhibited strong learning behavior, showing improved success rate and reduced repeated guesses over time.

Strategies

The HMM design was based on discrete emissions over the 26-letter alphabet. A separate HMM was trained for each word length, which allowed the model to capture structural letter dependencies specific to different word sizes. The HMM was trained using the Baum–Welch algorithm (an EM variant), which refined transition and emission probabilities. The RL agent used Q-learning on top of HMM predictions. The state representation combined the current masked word, the set of guessed letters, and the number of wrong guesses. This encoding balances simplicity and informativeness. Rewards were designed to reflect hangman outcomes: +10 for correct guesses, –10 for incorrect guesses, –2 for repeated guesses, and +100 for completing a word. This shaping provided a meaningful gradient for policy learning.

Exploration

Exploration versus exploitation was managed with an epsilon-greedy policy and exponential decay. Initially, epsilon was set to 1.0, encouraging broad exploration. Over time, epsilon decayed by 0.995 per episode down to a minimum of 0.01, promoting exploitation of learned strategies. Additionally, letter probabilities from the HMM provided a probabilistic prior that helped the agent make more informed exploratory choices rather than purely random ones.

Future Improvements

If given another week, I would focus on three main improvements: 1. Introduce a deep Q-network (DQN) to better generalize across states with similar masked patterns. 2. Augment the corpus with synthetically generated words to expand linguistic diversity and improve robustness. 3. Implement adaptive exploration — for example, decaying epsilon faster for shorter words and slower for longer ones. Additionally, incorporating subword models or contextual embeddings (like character-level RNNs) could further improve letter prediction accuracy.