Analysis Report: Hybrid HMM-RL Hangman Agent

Team-10 | ML SEM : 5 | 2025

Members : Lohit J : PES2UG23AM054
Chandan B L : PES2UG23AM021
Ganesh Jayadev : PES2UG23AM038
Banger Atif Ahmed : PES2UG23AM019

This report analyzes the architecture, strategies, and performance of the hybrid Hangman agent from the ML_HACKATHON_Team10.ipynb notebook. The agent's core design is a novel combination of a statistical language model (a Hidden Markov Model) and a strategic machine learning model (a Deep Q-Network).

1. Key Observations

- **Successful Learning:** The training logs clearly show the agent's "Average Reward" steadily increasing over 10,000 episodes, climbing from a negative value (e.g., -2.53) to a consistently positive one (e.g., +6.63). This confirms the agent's RL policy is successfully learning and improving.
- **Modest Final Performance:** Despite successful training, the final evaluation on 2,000 test words shows a success rate of only **28.1%**.
- **High Error Rate:** The most critical observation is the **5.36 average wrong guesses** per game. With a 6-guess limit, this means the agent is operating on the brink of failure in almost every game, whether it wins or loses. It is not an efficient player.
- **Sophisticated HMM:** The HMM is not a simple frequency counter. It is a collection of **separate, specialized models, one for each word length**. This is a strong design choice, as the statistical properties of 5-letter words are very different from 12-letter words.
- **Hybrid State Vector:** The RL agent's "state" is not just the game board. It's a concatenated vector of [**Game State** (board, guessed letters, wrong guesses) + **HMM Probabilities** (the full 26-letter probability vector)]. This is a form of feature engineering that allows the RL agent to "see" what the statistical expert is thinking.

2. Strategies

The agent's strategy is a two-part system that defines the value of guessing a letter (Q) as the sum of its statistical probability (P) and its strategic advantage (A).

Final Guess Value (Q) = HMM Probability (P) + DQN Advantage (A)

1. The Statistical Baseline (HMM Strategy):
   The HMM (AdvancedHMMHangman) acts as the statistical baseline, calculating the probability P for each letter. It does this by combining four distinct pieces of evidence:
   - **Positional Probability:** The likelihood of a letter appearing at a specific position (e.g., 'q' is almost always at the start).
   - **N-gram Context:** Bigram and trigram probabilities, which help guess common sequences like _ h _ -> 't' or i n _ -> 'g'.
   - **Pattern Matching:** It filters its dictionary for all words that match the current pattern (e.g., c o _ _ _ t e r) and analyzes the frequency of letters in the blank spots.
   - **Global Frequency:** The general, overall frequency of letters in the entire corpus.
2. The Strategic Correction (DQN Strategy):
   The DQN (RLHangmanAgent) learns the "Advantage" A. This A value is a correction or adjustment to the HMM's probabilities. It represents the long-term, learned strategic value of an action that the statistical model can't see.
   - **Example:** The HMM (P) might say 'e' has a 30% probability and 't' has a 20% probability. But the DQN (A) might have learned that in this *specific situation*, guessing 't' is strategically better (perhaps it reveals more information or avoids a common trap). It might assign A(t) = +0.5 and A(e) = -0.2.
   - **Final Decision:** The agent would calculate:
     - Q(e) = 0.30 + (-0.2) = 0.10
     - Q(t) = 0.20 + 0.5 = 0.70
   - ...and it would strategically choose 't', even though 'e' was statistically more probable.

3. Exploration vs. Exploitation Trade-off

The agent manages this trade-off using a **Guided Epsilon-Greedy** policy.

- **Framework:** It uses a standard epsilon variable that decays over time. At the start, epsilon is high (e.g., 1.0) so the agent explores. As training progresses, epsilon decreases, and the agent exploits its knowledge more often.
- Exploitation (When random > epsilon):
  The agent acts "greedily" on its learned strategy. It calculates the Q = P + A for every letter and chooses the letter with the highest Q-value. This is the agent's best-known move.
- Exploration (When random <= epsilon):
  This is a key part of the design. The agent does not pick a uniformly random letter (e.g., 'x', 'j', 'z'). That would be incredibly inefficient. Instead, it performs guided exploration. It takes the probability distribution P from the HMM and samples from it. This means its "exploratory" guesses are still smart—it's exploring among the most statistically likely letters. This is far more efficient than random exploration.

4. Future Improvements

1. **Improve the HMM Foundation:** The hybrid model is only as good as its P value. The agent's high error rate suggests the HMM is often wrong.
   - **Action:** Use a much larger and more diverse **corpus** (e.g., millions of words from public sources) to train the HMMs. A more accurate P value will give the DQN a better baseline to work from.
2. **Rethink the Reward Structure:** The agent is not penalized enough for wrong guesses.
   - **Action: Increase the penalty** for a wrong guess (e.g., from -1 to -2) and for losing (from -5 to -10). This "reward shaping" would more strongly discourage the agent from making the 5.36 wrong guesses it currently averages.
3. **Use a Better RL Architecture:** The current model uses a standard DQN to learn the Advantage. A more advanced architecture is a perfect fit here.
   - **Action:** Replace the DQN with a **Dueling DQN**. A Dueling DQN's architecture is specifically designed to separate the estimation of the *state's value* from the *advantage of each action*. This aligns perfectly with the Q = P + A design and would likely lead to a more stable and accurate A value.
4. **More Training:** 10,000 episodes is a good start, but given the 28.1% success rate, it is clearly not enough. The agent's reward was still climbing.
   - **Action:** Train the agent for **50,000 or 100,000 episodes** and adjust the epsilon_decay rate to be much slower, allowing for a longer exploration phase.