📝 Hangman Agent Analysis Report

GROUP – 3 SECTION – A[AIML]

Aditya Venkatesh SRN: PES1UG23AM024

Abhay H Bhargav SRN: PES1UG23AM008

Kushal Nayak M SRN: PES1UG24AM806

Aaditya Singh SRN: PES1UG23AM002

This report details the design, training, and evaluation of the Hangman agent implemented using a hybrid **Q-Learning and Statistical (HMM) approach**.

🧐 Key Observations

The final evaluation on the 2,000-word test set yielded a **negative final score of -18465.0**, indicating poor performance against the specified scoring metric.

Metric	Value	Interpretation
Success Rate	18.75%	The agent solved only 375 out of 2,000 games.
Total Wrong	11,193	This averages to **~5.6 wrong guesses per game** (out of a max of 6), suggesting the agent often loses after utilizing nearly all its lives.
Total Repeats	0	The agent successfully avoids repeating guesses during the greedy evaluation phase.

Most Challenging Parts

1. **Simplified State Abstraction:** The Q-learning state space (clamped_length, lives_left, revealed_count) is highly compressed. While necessary for a tabular Q-table, this abstraction likely loses critical information (like the specific positions of revealed letters or the co-occurrence of letters) that is essential for effective Hangman strategy, limiting the RL component's learning potential.
2. **Heavy HMM Reliance:** The agent is configured to heavily favor the statistical HMM component (mix_hmm = 0.9). While the HMM is a strong baseline, the Q-learning component, which is meant to learn *when* to deviate from the HMM based on the game state (e.g., specific word lengths, number of lives), has minimal influence.

🎯 Strategies: HMM and RL Design

HMM Design Choices

The agent uses a hybrid statistical model (hmm_letter_distribution) combining positional probability with **candidate filtering**.

- **Positional Probability:** Pre-computed letter counts are maintained for every position across different word lengths in the corpus, smoothed with an additive factor ($\alpha=1.0$).

- **Candidate Filtering:** The core statistical strategy is:
  1. **Filter** the corpus to only include words matching the current known pattern and excluding known wrong letters.
  2. If candidates exist, calculate letter counts from the letters in the **unknown positions** of the *candidate words*. This provides highly contextual, conditional probabilities.
  3. If no candidates exist (rare but possible), it falls back to a simple aggregate of positional probabilities across all unknown slots.

RL State and Reward Design

State Design

The agent uses **Tabular Q-Learning** with a highly abstracted state space defined by `state_key`:

$$\text{State} = (\text{clamped\_length}, \text{lives\_left}, \text{revealed\_count})$$

- **clamped_length**: $\min(\text{length of word}, 20)$. (Reduces table size for very long words).
- **lives_left**: 0 to 6.
- **revealed_count**: The count of letters revealed so far.

Rationale: This design creates a manageable, discrete state space suitable for tabular Q-learning (approximately $20 \times 7 \times 20 \approx 2800$ possible states), making convergence feasible within a limited number of episodes.

Reward Design

The reward function is dense and sparse, designed to shape behavior:

Action	Reward Value	Rationale
Correct Guess	$+0.5 + 0.2 \times (\text{new reveals})$	**Dense/Immediate.** Encourages correct guesses and scales the reward based on **information gain**.
Wrong Guess	-1.0	**Dense/Immediate.** Penalizes the loss of a life more than an incorrect but informative guess.
Repeat Guess	-0.5	**Dense/Immediate.** Penalizes exploration of an already-guessed letter.
Game Win	$+5.0$	**Sparse/Terminal.** Large bonus for solving the puzzle.
Game Loss	-2.0	**Sparse/Terminal.** Penalty for running out of lives.

💡 Exploration Management

The agent uses the **ϵ-greedy** policy to manage the exploration vs. exploitation trade-off, integrated with a statistical bias:

1. Hybrid Action Score: The action selection is based on a weighted mix of the learned Q-values and the statistical HMM scores:

   $$\text{Score} = (1.0 - \text{mix\_hmm}) \times \text{Normalized } Q + \text{mix\_hmm} \times \text{HMM Scores}$$

   With mix_hmm set to 0.9, the agent primarily exploits the statistical knowledge of the HMM.

2. **Decaying ϵ:**
    - **High Start:** ϵ starts at **0.9** (high exploration) to populate the Q-table and discover state transitions.
    - **Decay:** ϵ decays over time ($\epsilon_{decay}=0.9993$) to shift towards exploitation.
    - **Minimum ϵ:** ϵ_{min} is set to **0.1** to ensure continuous, minimal exploration (e.g., to discover better Q-values for rare states).
3. **Curriculum Learning:** The training starts with shorter words (length 3-8) for 5,000 episodes before expanding to the full corpus for another 5,000 episodes. This is a form of **Shaped Exploration**, ensuring the agent learns good initial policies on simpler problems before tackling the complexity of all word lengths.

🚀 Future Improvements

If given more time, the following improvements would be prioritized:

1. **Enriched RL State Representation (Move to DQN):**
    - The current tabular state is too simple. A **Deep Q-Network (DQN)** should be implemented to handle a richer state, such as an input vector representing the word pattern, the set of wrong guesses, and the number of lives. This would allow the agent to learn context-specific strategies based on the *actual* pattern (`_pp_e`), not just the *count* of revealed letters.
2. **Advanced HMM Integration (Bigrams/Trigrams):**
    - The statistical model should be upgraded to include **Bigram or Trigram positional frequency counts**. This would significantly improve accuracy, especially when the candidate set is small, by exploiting letter co-occurrence statistics (e.g., an 'e' at the end is often preceded by 'c', 't', or 'l').
3. **Dynamic HMM/Q-Value Mixing:**
    - Instead of a fixed `mix_hmm = 0.9`, the mixing parameter should be **dynamic**. For instance, the agent could rely more on the HMM initially (high mix_hmm) and then rely more on the learned Q-values (lower mix_hmm) as the game progresses and the specific state key is encountered more often.
4. **Reward Shaping for Information Gain:**
    - Adjust the reward to explicitly reward the **reduction in candidate set size** (entropy reduction), which is a direct measure of information gain, rather than just the number of reveals. This would encourage guesses that are most effective at disambiguating the secret word.