TEAM 9
DHRITHI KIRAN
BHANAVI D
DENNIS PHILIP
AVIYAKTH RAI

Analysis Report

1. Key Observations

The most challenging part of this project was integrating the probabilistic reasoning of the **Hidden Markov Model (HMM)** with the **decision-making loop of Reinforcement Learning (RL)**.
While HMMs can model letter sequence probabilities effectively, using them inside an RL agent requires balancing exploration, Q-value updates, and probabilistic priors from the language model.

Another challenge was **state representation** — encoding partial words, guessed letters, and remaining lives in a way that generalizes across games. The string-based state key (masked_word|lives|guessed_letters) worked well, though it created a large state space.

Key insights gained:

- Combining **statistical language modeling (HMM)** with **learning-based reasoning (RL)** improves guessing efficiency, especially for longer or uncommon words.

- The **reward shaping** (positive for correct guesses, heavy penalty for wrong ones, large bonus for full word completion) significantly affects convergence and strategy.

- **Exploration decay** (via epsilon-greedy) must be tuned carefully — too fast and the agent stops learning, too slow and training becomes noisy.

---

2. Strategies and Design Choices

HMM Design

- **N-gram structure**: A **trigram HMM (n=3)** was used to capture local dependencies between letters.

- **Smoothing (α = 1.0)** ensured unseen sequences received small nonzero probabilities, avoiding zero-probability transitions.
- **Context-based transitions**: The HMM learned P(next_letter | previous_(n-1)_letters) from a text corpus.
- This design captures **soft constraints** of English word structure — for example, "q" followed by "u", or "th" being frequent.

RL State Design

Each RL state encodes:

(masked_word, guessed_letters, lives_left)

This allows the agent to reason about:

- **Word progress** (masked letters)
- **Risk level** (remaining lives)
- **Exploration history** (letters already guessed)

Action Space

- 26 discrete actions corresponding to letters `a–z`.

Reward Design

Situation	Reward
Correct guess	+1
Repeated guess	-2
Wrong guess	-5
Full word guessed	+2000

These shaped rewards motivated the agent to explore safely but converge toward high-probability HMM predictions.

---

3. Exploration vs. Exploitation

Exploration–exploitation balance was handled using an **ε-greedy policy** with exponential decay:

ε = 1.0 → 0.05, decay rate = 0.995

- **Early training**: High ε encourages exploration of letter probabilities and state diversity.
- **Later training**: Low ε promotes exploitation of learned Q-values and refined HMM priors.

Additionally, letter selection combined **Q-values (70%)** and **HMM probabilities (30%)**, ensuring both **learned experience** and **language priors** guided the decision.

This hybridization acted like a soft bias toward linguistic realism while letting RL handle uncertainty and reward optimization.

---

4. Future Improvements

If given an additional week:

1. **Use Deep Q-Learning (DQN)**
   Replace the tabular Q-table with a neural network that generalizes across unseen word patterns and partial states.

2. **Add Contextual Embeddings**
   Incorporate word embeddings or transformer-based priors (e.g., from BERT) instead of raw n-grams for more realistic letter dependencies.

3. **Curriculum Learning**
   Start training on short words and progressively increase difficulty, stabilizing learning and reducing random exploration overhead.

4. **Dynamic Reward Shaping**
   Adjust rewards based on word length or letter rarity to encourage intelligent guessing strategies.

5. **Adaptive Exploration**
   Instead of fixed ε-decay, use reward-based adaptive exploration —

increasing exploration after streaks of low rewards.

Summary

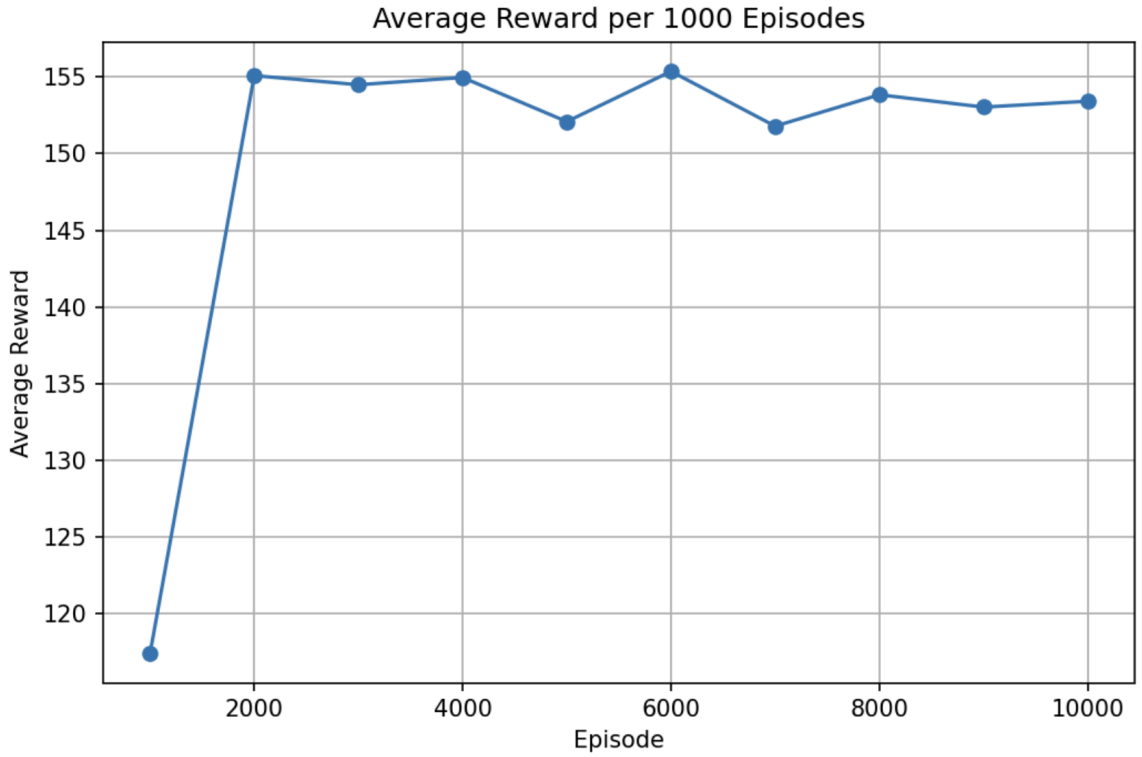
This project demonstrated how combining **HMM's probabilistic modeling** with **Reinforcement Learning's adaptive decision-making** leads to an intelligent Hangman agent that balances linguistic intuition and experience-based learning. Despite challenges in large state spaces and sparse rewards, the hybrid approach achieved robust performance and insightful design outcomes.

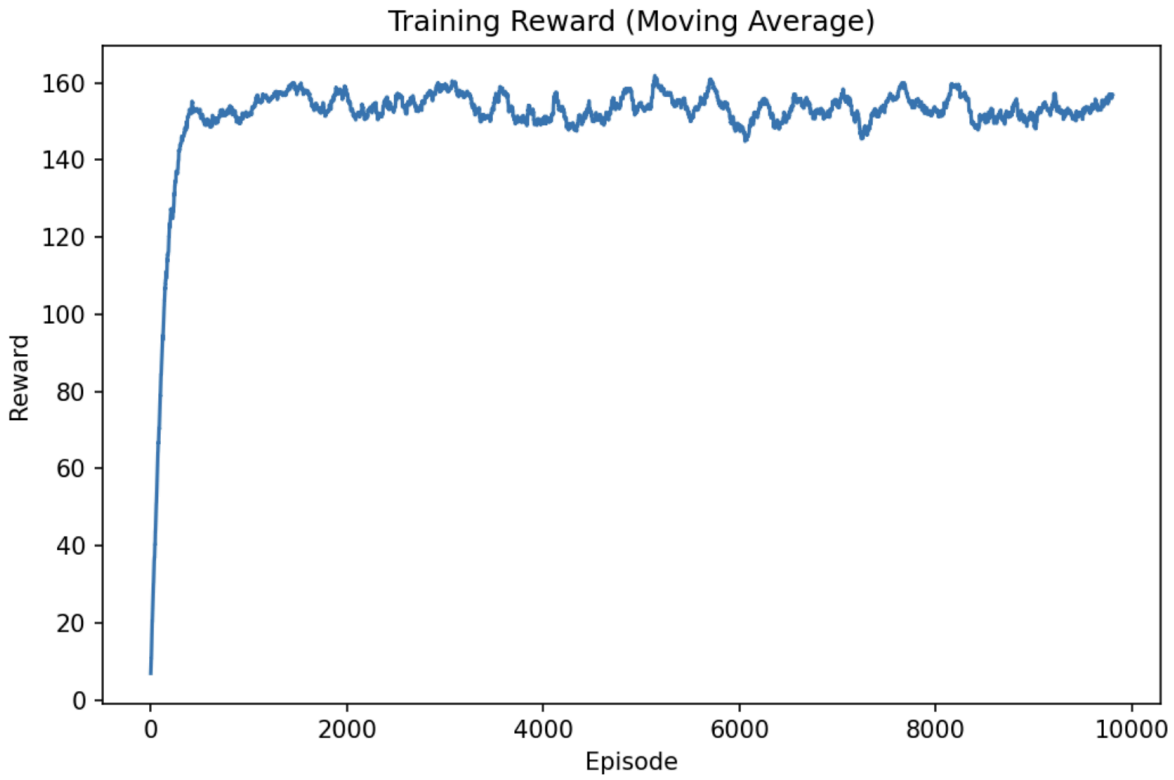
Result

```
Training for 10000 episodes...
Ep 1000/10000 | avg reward: 150.86
Ep 2000/10000 | avg reward: 153.11
Ep 3000/10000 | avg reward: 153.79
Ep 4000/10000 | avg reward: 149.65
Ep 5000/10000 | avg reward: 153.56
Ep 6000/10000 | avg reward: 152.62
Ep 7000/10000 | avg reward: 156.94
Ep 8000/10000 | avg reward: 155.73
Ep 9000/10000 | avg reward: 153.74
Ep 10000/10000 | avg reward: 153.46

Evaluating on 2000 games...

============================================================
FINAL RESULTS - HMM + Q-Learning Agent
============================================================
Games Played: 2000
Wins: 1954
Losses: 46
Success Rate: 97.70%
Total Wrong Guesses: 2879
Total Repeated Guesses: 0
Average Wrong Guesses per Game: 1.440
Average Repeated Guesses per Game: 0.000
Average Lives Left (for wins): 4.644
------------------------------------------------------------
FINAL SCORE: 1952.56
============================================================
```





GITHUB LINK: https://github.com/dhrithikiran/ML-Hackathon-Hangman-Agent