UE23CS352A: Machine Learning Hackathon

Hackman

Team number:14

Members:

Abhinav Mekala Babu: PES2UG23CS338

Mrinal Pandey:PES2UG23CS353

Nevin Mathew Thomas:PES2UG23CS381

Nehal G: PES2UG23CS380

1. Key Observations

Training the Hangman reinforcement learning agent with an embedded Hidden Markov Model (HMM) produced clear and consistent patterns throughout the experiment.
The HMM learned recognizable language tendencies from the 50,000-word dataset. The entropy results (~2.97 for the starting letter distribution and ~2.92 for unigram frequencies) showed that the model captured typical English letter variation rather than overfitting to a few common letters.

When the HMM was combined with the Q-learning agent, the system started making more reasonable guesses earlier in training. Instead of guessing letters completely at random, it appeared to use some understanding of which letters were likely to occur in certain positions. This suggests that the HMM gave the agent a stronger initial sense of structure before any reinforcement learning took effect.

Throughout training, the environment remained difficult because feedback was limited and delayed. The agent only learned from each guess, and most actions gave small or negative rewards. The reward graph showed frequent ups and downs — total reward ranged roughly between −10 and +20 per episode — which reflects the unpredictable nature of Hangman rather than model instability. Over time, rewards slowly moved closer to zero, suggesting small but steady improvement.

The success-rate plot stayed between about **30% and 40%** across training. By the end of 2,000 episodes, the agent reached a final success rate of **35.45%**. This shows that while learning occurred, progress leveled off after a certain point.Importantly, the agent never repeated guesses (average repeated guesses = 0), showing that it reliably tracked its previous actions.

Overall, the results suggest that the HMM provided a useful foundation for guiding letter choices, while the reinforcement learning component helped refine those choices through repeated gameplay. The combination produced an agent that learned consistent patterns but still faced limits due to the sparsity of rewards and partial visibility of the game state.

2. Strategies and Design Decisions

Hidden Markov Model (HMM)

- **Hidden States:** each word position is paired with a possible letter:(position, letter).
- **Transitions:** bigram probabilities P($letter_{i+1}$ | $letter_i$) to encode likely letter sequences (“th”, “an”, “er”, etc.).
- **Emissions:** deterministic based on whether a letter is visible ('a') or hidden ('_').
- **Positional priors:** the word is divided into ten relative buckets (0–1.0) to represent where letters typically appear (e.g., vowels early, “e” or “y” near the end).

  This structure gave our model a solid linguistic grounding. Even before introducing reinforcement learning, we found that the HMM could already rank letters in a way that roughly matched human intuition : for example, prioritizing common vowels and frequent consonants like *t* or *n* while avoiding rarer letters such as *q* or *z*.

  We chose this design because we wanted the agent to start with a realistic understanding of how English words are formed, rather than learning from scratch through random guessing. By modeling letter transitions and position-based patterns, we gave the HMM the ability to capture simple word structure , things like how certain letters often follow each other or where vowels tend to appear. This made the model’s predictions more interpretable and gave the reinforcement learning component a more informed starting point.

  In short, we used the HMM to provide linguistic context, allowing the RL agent to focus on strategic decision-making — such as choosing the best next guess , instead of having to rediscover basic language rules through trial and error.

Reinforcement Learning Agent

- **State:** (masked_word, guessed_letters, wrong_guesses)
  captures visible pattern, exploration history, and penalty pressure.
- **Actions:** 26 possible letter guesses.

Outcome	**Reward**
Correct letter	+1
Complete win	+10
Incorrect letter	−1
Game lost (max wrongs)	−5
Repeated guess	−2

- **Decision Fusion:**
  Instead of relying solely on Q-values, actions are scored as

$$0.3 \times Q(s,a) + 0.7 \times P_{\text{HMM}}(a)$$

We chose to give more weight to the HMM's probabilities because they were based on real language patterns, while still allowing the RL agent to adjust its behavior through game feedback. This mix worked well : the HMM kept the agent's guesses logical and consistent, and the Q-learning part helped it adapt over time to what actually led to wins.

3. Exploration vs. Exploitation

An ε-greedy strategy managed exploration:

- **Initial ε = 0.5** encouraged broad exploration.
- **Exponential decay (ε × 0.999 per episode)** reduced it to about **0.07** by the end of training.

Early in training, random guesses dominated, helping the agent gather varied state–reward experiences.As ε decayed, exploitation took over, letting the agent rely on the HMM-guided policy it had refined.

The smooth oscillation in success rates matches this transition — exploration spikes occasionally dropped performance temporarily, but exploitation gradually stabilized results around the 35 % mark.

This progression mirrors human learning in Hangman: at first, you try random letters, but over time you default to educated guesses like vowels or frequent consonants. The agent essentially learned a similar intuition.

4. Future Improvements

If I had another week, several meaningful upgrades would be next:

1. **Deep Q-Network (DQN)** – replace the tabular Q-table with a neural net so the agent can generalize across unseen word patterns and lengths.
2. **Adaptive Reward Shaping** – dynamically scale rewards based on the information gained (e.g., larger reward for revealing rare letters like ‘q’ or ‘z’).
3. **Higher-Order HMM** – add trigram or syllable-based transitions to better represent natural English spelling structures.
4. **Curriculum Training** – start with short words, then gradually include longer and more complex ones to smooth learning difficulty.
5. **Dynamic ε-Scheduling** – adjust exploration not only by time but also by reward variance (increase ε if learning plateaus).
6. **Visualization Tools** – heatmaps of posterior letter probabilities per position to better interpret and debug the HMM’s internal logic.

5. Summary of Final Metrics

Metric	Value
Final Success Rate	**~34.50 %**
Avg. Wrong Guesses	**~4.52 per game**
Avg. Repeated Guesses	**0.00**
Avg. Reward per Episode	**~ −1.17**
Training Episodes	**2000**
Final ε	**0.068**
HMM Entropy (start / unigram)	**2.974 / 2.916**