PES UNIVERSITY

Department of CSE (AI & ML)

Analysis Report-ML Hackathon

MACHINE LEARNING

UE23CS352A

Name	**P PAVAN SHANBHAG PURUSHOTHAM RANJAN C POOJA AMMALAJERI**
SRN	**PES1UG23AM199 PES1UG23AM221 PES1UG23AM232 PES1UG23AM203**
Section	**D**
Department	**CSE(AI&ML)**
Campus	**RR**
Date	**03-11-2025**

1. Key Observations

Designing an intelligent Hangman agent presented multiple challenges related to uncertainty, exploration, and sparse rewards.
The most difficult parts included:

- Balancing probabilistic reasoning with decision learning.
  The HMM provides letter probabilities but doesn't directly optimize for game outcomes, while Q-learning learns from rewards but needs structured guidance from the HMM.
- State explosion.
  Since Hangman words vary in length and the state space includes pattern + guesses + lives + probabilities, a naive Q-table could grow exponentially. Compacting the state with rounded HMM probabilities and a binary guessed-mask helped reduce dimensionality.
- Reward shaping.
  Designing a reward function that both encourages efficiency (fewer wrong guesses) and punishes wasteful repetition required tuning.
- Epsilon decay tuning.
  Too much exploration caused excessive wrong guesses, while too little hindered learning early on. A slow multiplicative decay gave the best balance.

Through experimentation, the system achieved a stable learning curve, gradually improving the reward per episode and reducing repeated guesses.

2. Strategies and Design Choices

2.1 Hidden Markov Model (HMM)

- **Structure:**
  A bigram HMM was implemented from scratch, where:
  - Hidden states represent the sequence of letters.
  - Emissions correspond directly to the same letters (simplified since Hangman provides partial observations of characters).
  - Transition probabilities are estimated from the 50,000-word corpus, capturing character-to-character dependencies (e.g., 'q' → 'u').
- **Computation:**
  The agent uses forward–backward inference to estimate posterior probabilities of each letter at each blank position, conditioned on the revealed letters and previous guesses.
- **Purpose:**
  The HMM acts as a language prior, predicting which letters are most plausible given the masked pattern. This prevents the agent from guessing rare letters that don't fit English orthographic patterns.

2.2 Reinforcement Learning Environment

- **State Definition:**
  Each state includes:
  1. The current masked pattern.
  2. The set of guessed letters (binary 26-length mask).
  3. The number of remaining lives.
  4. The HMM-derived probability vector for all letters.

This combination gives both syntactic and probabilistic cues for decision-making.

- **Actions:**
  Any letter from the English alphabet that hasn't yet been guessed.
- **Reward Function:**

Outcome	Reward
Correct guess	+2 per hit (+1 per revealed occurrence)
Wrong guess	−5
Repeated guess	−2
Full word solved	+50
Game lost	−10

- This reward structure heavily incentivizes correctness and efficiency, mirroring the final evaluation formula**.**

2.3 Q-Learning Agent

- **Algorithm:**
  A tabular Q-learning setup with:
  - Learning rate α = 0.3
  - Discount factor γ = 0.95
  - Epsilon-greedy exploration starting at 0.6 and decaying to 0.05
- **Policy:**
  When exploiting, the agent selects the letter that maximizes
  Q(state, action) + 1.5 * P_HMM(letter)
  blending learned experience with HMM prior knowledge.
- **Training Loop:**
  8,000 episodes were run over random words from the training corpus.
  Rewards per episode and epsilon decay were tracked for analysis.

3. Exploration vs. Exploitation

The exploration–exploitation trade-off was managed using a multiplicative epsilon decay strategy:

$$\epsilon_t = \max(\epsilon_{\text{end}}, \epsilon_{\text{start}} \times \text{decay}^t)$$

This allows:

- Early exploration to discover valuable letter–pattern relationships.
- Gradual exploitation as learning stabilizes.

Visualization confirmed that reward variance decreased as epsilon declined, showing more consistent policy performance over time.

4. Evaluation & Learning Insights

4.1 Metrics Collected

- Success Rate (SR) across 2,000 evaluation games
- Average Wrong Guesses per game

- Average Repeated Guesses
- Final Score using:

$$\text{Score} = (SR \times 2000) - 5 \times \text{Wrong} - 2 \times \text{Repeat}$$

4.2 Observed Patterns

- The reward per episode curve shows steady improvement, indicating successful convergence.
- Epsilon decay visualization confirms exploration reduction over training.
- The smoothed reward plot flattens toward a positive plateau, showing policy stability.
- The evaluation summary bar chart highlights a strong success rate with low average wrong and repeated guesses.

5. Future Improvements

1. Deep Q-Network (DQN):
   Replace the Q-table with a neural approximator capable of handling high-dimensional continuous HMM probability vectors.
2. Context-aware HMMs:
   Train length-specific HMMs (e.g., for 4-, 5-, 6-letter words) or include trigram transitions for richer letter dependencies.
3. Reward Shaping via Partial Progress:
   Reward proportional to the fraction of word revealed, encouraging incremental discovery.
4. Curriculum Learning:
   Start with short words and gradually introduce longer, complex ones to stabilize learning.
5. Adaptive Exploration:
   Implement Boltzmann exploration instead of fixed epsilon decay for more dynamic action selection.

6. Key Takeaways

- Integrating an HMM prior with Q-learning decision-making significantly improves guessing efficiency.
- The hybrid system balances linguistic knowledge (from corpus statistics) with adaptive strategy learning (from experience).
- Visualizations demonstrate tangible learning progression and clear exploration control.
- The resulting agent achieves strong performance aligned with the evaluation formula, showing intelligent Hangman behavior.

GitHub Link:

https://github.com/pes1ug23am199/ML_Hackathon.git