Hangman Q-Learning Agent with HMM Oracle

Technical Report

Team Information

- **Team Number:** 1
- **Team Members:**
    1. Sarana Gnanojval (PES1UG23AM274)
    2. Siddharth (PES1UG23AM303)
    3. Shreyas S Mallappa (PES1UG23AM296)
    4. S Sriram (PES1UG23AM248)

Executive Summary

This project presents a novel approach to playing Hangman by combining Q-Learning reinforcement learning with Hidden Markov Model (HMM) probabilistic guidance. The developed agent achieved a **41.30% success rate** on a test set of 2000 words, representing a **23.60% improvement** over the baseline accuracy of 17.70%. The agent successfully minimized repeated guesses (0 repeated guesses in evaluation) while managing wrong guesses strategically.

---

1. Introduction

1.1 Problem Statement

Hangman is a word-guessing game where a player must identify a hidden word by guessing letters within a limited number of incorrect attempts. The challenge lies in:

- Maximizing the probability of correct letter guesses
- Minimizing wrong guesses (penalty: 5 points each)
- Avoiding repeated guesses (penalty: 2 points each)

- Achieving the highest possible score based on: `(Success Rate × 2000) - (Wrong Guesses × 5) - (Repeated Guesses × 2)`

1.2 Objectives

1. Develop a reinforcement learning agent capable of learning optimal Hangman strategies
2. Integrate probabilistic language modeling to guide decision-making
3. Achieve significant improvement over baseline random or heuristic approaches
4. Minimize penalties from wrong and repeated guesses

---

2. Methodology

2.1 System Architecture

The solution comprises three main components working in tandem:

2.1.1 HMM Oracle

A probabilistic language model that provides statistical guidance for letter selection.

Key Features:

- Trained on a large corpus of words to learn letter transition patterns
- Maintains separate models for each word length
- Uses Forward-Backward algorithm for probability calculation
- Provides probability distribution over remaining letters given:
    - Current masked word state
    - Previously guessed letters

Mathematical Foundation:

- **Initial State Distribution (π):** Probability of each letter being the first character
- **Transition Matrices (A):** Position-dependent letter-to-letter transition probabilities
- **Forward-Backward Algorithm:** Computes marginal probabilities for each unknown position

$\alpha_t(j) = P(\text{observations up to } t, \text{ state } j \text{ at } t)$
$\beta_t(j) = P(\text{observations after } t \mid \text{state } j \text{ at } t)$
$P(\text{letter at position } t) \propto \alpha_t(j) \times \beta_t(j)$

2.1.2 Hangman Environment

A simulation environment implementing the game logic for training and evaluation.

State Components:

- Remaining lives (0-6)
- Masked word representation (fixed-length encoding)
- Guessed letters (boolean tuple for all 26 letters)
- Top-3 HMM recommendations (discretized for state space reduction)

Reward Structure:

- +1.0 per correct letter revealed
- +10.0 bonus for winning the game
- -1.0 penalty for repeated guess
- -2.0 penalty for wrong guess
- -10.0 penalty for losing the game
- -0.01 per step to encourage efficiency

2.1.3 Q-Learning Agent

A tabular reinforcement learning agent that learns optimal action policies.

Core Algorithm:

$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \cdot \max_{a'} Q(s',a') - Q(s,a)]$

Where:

- $\alpha = 0.05$ (learning rate)
- $\gamma = 0.99$ (discount factor)
- $\varepsilon = 1.0 \rightarrow 0.05$ (exploration rate with decay = 0.9992)

Action Selection Strategy: During exploitation, the agent combines Q-values with HMM probabilities multiplicatively:

$score(a) = (Q_{normalized}(a) + \varepsilon) \times (HMM_{prob}(a)^w + \varepsilon)$

Where:

- $w = 0.5$ (exploration weight controlling HMM influence)
- $\varepsilon = 1e\text{-}6$ (small constant to avoid zero scores)

2.2 State Space Design

A critical challenge in tabular Q-Learning is managing the state space explosion. Our approach:

1. **Masked Word Encoding:** Pad/truncate to fixed length (20 characters)
2. **Guessed Letters:** Binary encoding (26-bit tuple)
3. **HMM Guidance:** Include top-3 HMM recommendations as state features
4. **Lives Remaining:** Discrete value (0-6)

This representation creates a tractable yet informative state space that captures essential game dynamics.

2.3 Training Process

Hyperparameters:

- Training Episodes: 3000
- Learning Rate (α): 0.05
- Discount Factor (γ): 0.99
- Epsilon Decay: 0.9992
- Evaluation Frequency: Every 200 episodes

Training Procedure:

1. Initialize Q-table as empty defaultdict
2. For each episode:
    - Reset environment with random word
    - While game not over:
        - Select action using ε-greedy + HMM guidance
        - Execute action, observe reward and next state
        - Update Q-value using Q-learning rule
        - Decay exploration rate
3. Track metrics: rewards, win rate, wrong guesses, Q-table size

---

3. Results

3.1 Training Performance

Training Metrics (3000 episodes):

- **Episode Rewards:** Showed consistent upward trend, indicating successful learning
- **Win Rate:**
    - Started near 0%
    - Peaked around episode 1200
    - Stabilized at ~40-45% moving average
- **Wrong Guesses:** Decreased from ~6 average to ~4.8 average
- **Q-Table Growth:** Final size captured substantial state space coverage

3.2 Evaluation Results

Test Set Performance (2000 words):

Metric	Value
Success Rate	41.30% (826/2000)
Total Wrong Guesses	9,795
Total Repeated Guesses	0
Avg Wrong per Game	4.90
Avg Repeated per Game	0.00
Final Score	-48,149.00

3.3 Performance Improvement

Accuracy Comparison:

- **Baseline Accuracy:** 17.70%
- **Optimized Agent:** 41.30%
- **Improvement:** +23.60 percentage points (133% increase)

This represents more than doubling the success rate compared to the initial approach.

---

4. Analysis and Discussion

4.1 Strengths

1. **No Repeated Guesses:** Perfect elimination of repeated guess penalties through state tracking
2. **Significant Learning:** Clear improvement over 3000 episodes with stable convergence
3. **HMM Integration:** Probabilistic guidance effectively reduces search space and improves initial guesses
4. **Scalable Architecture:** Modular design allows component improvements independently

4.2 Limitations

1. **Negative Final Score:** High cumulative penalty from wrong guesses (-48,149)
    - Each wrong guess costs 5 points
    - 9,795 wrong guesses × 5 = 48,975 penalty points
    - Success bonus: 826 × 2000 = 1,652 points
    - Net effect is negative
2. **State Space Challenges:**
    - Fixed-length encoding may lose information for very long/short words
    - Q-table still grows large (potentially millions of states)
    - Generalization across similar states is limited
3. **Win Rate Plateau:** Agent struggles with harder words despite good performance on average

4.3 Key Insights

Why the Negative Score? The scoring formula heavily penalizes wrong guesses. Even with a 41.3% win rate, the agent makes ~4.90 wrong guesses per game on average:

- Losses (1174 games): Often use all 6 wrong guesses
- Wins (826 games): Still average ~2-3 wrong guesses

HMM Contribution: The multiplicative combination of Q-values and HMM probabilities proved effective:

- Early training: HMM provides strong guidance when Q-table is sparse
- Later training: Q-values dominate as learned experience accumulates
- Evaluation: Balanced combination leverages both statistical priors and learned strategy

Learning Dynamics:

- Rapid initial improvement (episodes 0-1000)
- Slower refinement phase (episodes 1000-2000)
- Stabilization and overfitting prevention (episodes 2000-3000)

---

5. Future Work

5.1 Immediate Improvements

1. **Deep Q-Networks (DQN):**
    - Replace tabular Q-table with neural network

- Better generalization across states
- Handle larger state spaces efficiently

5.2 Advanced Enhancements

1. **Ensemble Methods:**
    - Combine multiple HMM models (trained on different corpora)
    - Use word embeddings (Word2Vec, BERT) for semantic guidance
    - Implement letter frequency analysis specific to partial word patterns

---

6. Conclusion

This project successfully demonstrated the application of reinforcement learning to strategic game-playing through the development of a Hangman agent that combines Q-Learning with HMM probabilistic guidance. The agent achieved:

✅ **41.30% success rate** on unseen test words
✅ **133% improvement** over baseline accuracy
✅ **Zero repeated guesses** in evaluation
✅ **Stable learning** over 3000 training episodes

While the final score is negative due to the penalty structure, the agent learned meaningful strategies and significantly outperformed naive approaches. The modular architecture provides a strong foundation for future enhancements using deep learning and advanced RL techniques.

The integration of domain knowledge (HMM language modeling) with data-driven learning (Q-Learning) proved to be an effective strategy, demonstrating the value of hybrid approaches in solving complex sequential decision problems.

Hyperparameter Tuning Results

Parameter	Initial Value	Optimized Value	Impact
Learning Rate (α)	0.1	0.05	More stable convergence
Discount Factor (γ)	0.95	0.99	Better long-term planning
Epsilon Decay	0.995	0.9992	Slower exploration decay
Training Episodes	1000	3000	Better Q-table coverage

Exploration Weight	1.0	0.5	Balanced Q-HMM influence