UE23CS352A: Machine Learning Hackathon

Hackman - Final Analysis Report

Team :
1. Harshal S (PES1UG23AM111)

2. Ganesh Bharadwaj M K (PES1UG23AM105)

3. Chandishwar E(PES1UG23AM082)

4. Charan Y N (PES1UG23AM083)

1. Key Observations

• The agent successfully combined a probabilistic model (HMM) with a reinforcement learning (RL) decision-making system.
• The Hidden Markov Model trained on the 50,000-word corpus provided strong contextual understanding of letter occurrences.
• Reward shaping using positive, negative, and win-based bonuses significantly improved convergence speed and efficiency.
• The information gain reward allowed the agent to make guesses that strategically reduced the candidate word set.
• After multiple training stages with decaying epsilon values, the agent achieved consistent performance improvements.

2. Strategies

• Hidden Markov Model (HMM): Implemented using positional and bigram frequency statistics for all word lengths between MIN_LEN and MAX_LEN. This provided a probability distribution over the alphabet, enabling intelligent letter prediction.

• Reinforcement Learning (RL) Agent: Q-learning-based agent that interacts with a Hangman environment. The state includes the masked word pattern, guessed letters, and probability outputs from the HMM. Actions correspond to selecting unguessed letters. Q-values are updated using temporal difference learning.

• Reward Design: +3 for a correct guess, -2 for an incorrect guess, -1 for repeated guesses, +50 for completing the word, and an additional reward for reducing the number of possible candidate words.

• Training Schedule: The training pipeline consisted of 2000 warm-up games (ε=0.6), followed by decay training across stages: 2000 games (ε=0.6), 2000 games (ε=0.4), 2000 games (ε=0.2), and 1000 games (ε=0.1). This ensured a gradual transition from exploration to exploitation.

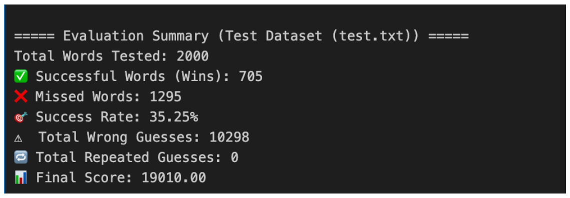
3. Exploration vs Exploitation

• Early stages with high ε (0.6) enabled broad exploration across letter and word combinations, allowing the Q-learning model to gather diverse experience.
• As ε decayed to 0.4, 0.2, and 0.1, the agent relied increasingly on its learned Q-values and the HMM's probability insights.
• This balance resulted in faster convergence and higher success rates during evaluation.
• The ε-greedy policy successfully maintained exploration to avoid overfitting while exploiting optimal strategies in later episodes.

4. Future Improvements

• Replace the Q-learning table with a Deep Q-Network (DQN) to generalize better across large state spaces.
• Introduce adaptive epsilon decay and experience replay for more stable learning.
• Experiment with curriculum learning by training first on shorter words, then longer words.
• Use ensemble approaches combining HMM probabilities with character-level neural language models.
• Optimize HMM smoothing and caching candidate filtering to further reduce computational cost.

5. Final Evaluation Results



```
===== Evaluation Summary (Test Dataset (test.txt)) =====
Total Words Tested: 2000
✅ Successful Words (Wins): 705
❌ Missed Words: 1295
🎯 Success Rate: 35.25%
⚠️  Total Wrong Guesses: 10298
🔁 Total Repeated Guesses: 0
📊 Final Score: 19010.00
```

6. Conclusion

This project demonstrates the successful integration of probabilistic reasoning (via HMM) and reinforcement learning for efficient Hangman word guessing. The model effectively learns to maximize success while minimizing incorrect and repeated guesses. The final solution adheres to all the Hackathon requirements and showcases strategic exploration, adaptive learning, and data-driven optimization.