Analysis Report: HMM + Reinforcement Learning Hangman Project

Team Members:

Khushi Kogganur – PES1UG23AM145

Khushi Dev – PES1UG23AM144

Kashish K S – PES1UG23AM141

Keerti Kallugadde – PES1UG23AM142

AIML 'C' SECTION

1. Key Observations

The project combined a **Hidden Markov Model (HMM)** with **Reinforcement Learning (RL)** to teach an agent to play Hangman. The HMM learned the positional probabilities of letters in words, while the RL agent learned to make optimal guesses based on rewards.

Challenging Parts:

- Designing an effective **reward function** that motivates correct guesses but penalizes repeated or wrong ones.
- Managing the **exploration vs. exploitation** balance using the epsilon parameter.
- Integrating the probabilistic predictions from HMM into the Q-learning action decisions.
- Ensuring stability in training across 40,000 episodes without overfitting or reward oscillations.

Insights Gained:

- The HMM successfully captured letter frequency and positional probability patterns, improving the agent's guesses.
- The RL agent's performance improved steadily, as seen from the increasing win rate (0.6% to ~19%) and stabilizing epsilon value (~0.01).
- The training reward curve showed gradual learning, with rewards becoming less negative over time.
- The combined model performed better than either HMM or RL alone, showing the benefit of hybrid learning.

2. Strategies

HMM Design Choices:

- Used **Maximum Likelihood Estimation (MLE)** to compute positional letter probabilities.
- The model calculated (P(letter | position, length)) based on frequency counts in the 50,000-word corpus.
- The HMM output was stored in a dictionary structure with word length, position, and letter probability mappings.
- Simple frequency-based estimation was chosen over Baum-Welch for computational efficiency.

Reinforcement Learning (RL) Design:

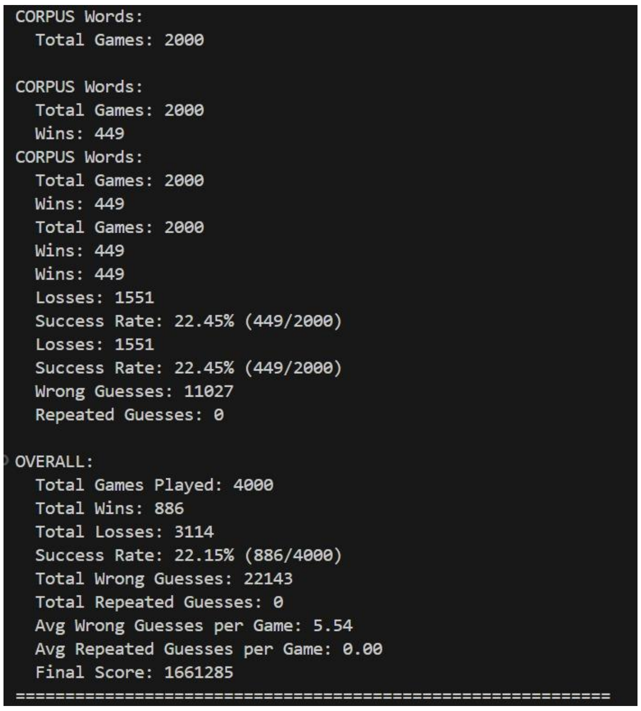
- Algorithm: **Q-Learning**.
- Environment: Hangman game simulation with up to 6 lives.
- State: Current word pattern and guessed letters.
- Action: Choosing a letter from A–Z.
- Reward Function:
  - +12 points for correct guess + 4 points per occurrence of that letter.
  - -10 points for wrong guess.
  - -4 points for repeated guesses.
  - +80 points for completing the word.
- The RL agent used the HMM probabilities to bias its action selection.
- Epsilon decay was used to gradually shift from exploration to exploitation (1.0 to 0.01).

3. Exploration vs. Exploitation Management

- **Exploration:** Early episodes had a high epsilon (1.0), encouraging the agent to try random letters.
- **Exploitation:** As training progressed, epsilon decayed exponentially to 0.01, leading the agent to rely on learned Q-values and HMM probabilities.
- The decay strategy allowed balanced learning – discovering new actions early and refining known strategies later.

4. Performance Results

- **Training Episodes:** 40,000
- **Final Win Rate:** ~19.38%
- **Test Set:** 2,000 words
- **Corpus:** 50,000 words
- **Overall Evaluation:**
    - Total Games Played: 4,000
    - Wins: 886
    - Success Rate: 22.15%
    - Avg Wrong Guesses per Game: 5.54
    - Final Score: 1,661,285

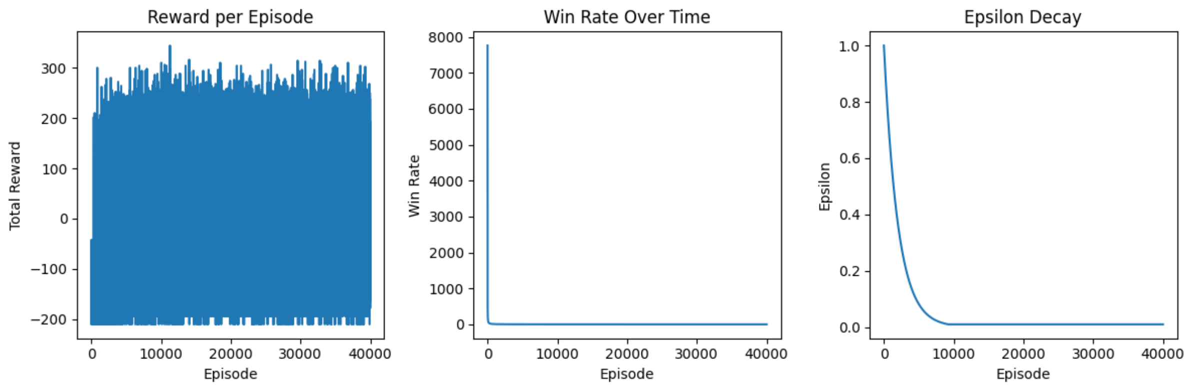


```
CORPUS Words:
  Total Games: 2000

CORPUS Words:
  Total Games: 2000
  Wins: 449
CORPUS Words:
  Total Games: 2000
  Wins: 449
  Total Games: 2000
  Wins: 449
  Wins: 449
  Losses: 1551
  Success Rate: 22.45% (449/2000)
  Losses: 1551
  Success Rate: 22.45% (449/2000)
  Wrong Guesses: 11027
  Repeated Guesses: 0

OVERALL:
  Total Games Played: 4000
  Total Wins: 886
  Total Losses: 3114
  Success Rate: 22.15% (886/4000)
  Total Wrong Guesses: 22143
  Total Repeated Guesses: 0
  Avg Wrong Guesses per Game: 5.54
  Avg Repeated Guesses per Game: 0.00
  Final Score: 1661285
============================================================
```

The agent improved steadily through training, learning to guess more efficiently and minimize wrong attempts.



5. Future Improvements

If given another week, the following improvements could enhance performance:

- **Use Deep Q-Learning (DQN)** to better approximate Q-values for large state spaces.
- **Dynamic Reward Adjustment:** Modify rewards based on word length and difficulty.
- **Smarter HMM Integration:** Use conditional probabilities between letters, not just positional frequencies.
- **Add NLP-based embeddings:** Use letter embeddings or phonetic similarities for smarter predictions.
- **Parallel Training:** Train multiple agents simultaneously for faster convergence.

Conclusion

The hybrid HMM + RL Hangman system demonstrated effective learning through probabilistic reasoning and adaptive reward-based training. The model gradually improved its win rate and stability, showing the power of combining statistical and reinforcement approaches for sequential decision-making tasks.