Analysis Report: HMM and Reinforcement Learning Agent for Hangman

TEAM ID:

NAME:DHEEKSHA	SRN:PES2UG23CS170
RAKSHITHA	PES2UG23CS134
SHREYA RAJ	PES2UG23CS136
CHETHANA KR	PES2UG23CS151

1. Introduction

This report details the strategies employed, key observations, and challenges encountered during the development of a Hangman-playing agent using a hybrid model incorporating a Character Hidden Markov Model (HMM) and a Reinforcement Learning (RL) agent specifically, Tabular Q-Learning (QL) or Deep Q-Network (DQN).

2. Key Observations and Challenges

2.1 Most Challenging Parts

• State Space Explosion (Tabular QL): The primary challenge for the Tabular Q-Learning approach is the vast, discrete state space. Each state represents a combination of the masked word, word length, and the set of guessed letters. Since there are 2^26 possible guessed-letter combinations, maintaining and training a Q table for all states becomes infeasible.

• Credit Assignment: Determining whether a correct guess is a good strategic move or a lucky one is difficult. Q-Learning mitigates this through temporal difference learning, but this requires millions of training episodes.

• Integration of HMM Features: Combining the HMM's continuous probabilistic outputs (26-letter probability vector) into the RL state was crucial. In Tabular QL, this feature is ignored in the key but still influences learning indirectly. In DQN, it serves as a continuous input vector, improving the agent's ability to generalize.

3. Key Insights Gained:

• HMM as a Feature Extractor: The HMM alone performs well as a heuristic solver, predicting likely letters from context. Integrating its output into the RL state significantly reduces learning complexity — the RL agent learns when to trust the HMM versus when to explore other letters.

• State Abstraction is Essential: For scalability, the masked word string must be represented more abstractly — e.g., by pattern (_A_B_) or length — instead of literal string keys. The current representation works but doesn't scale well.

4. Strategies and Design Choices

4.1 HMM Design

Goal: Model sequential dependencies between characters in English words.
Transition Probabilities: Estimate P(c_{t+1} | c_t) from word corpus.
Emission Probabilities: Model P(o_t | s_t), though here states and observations are both characters.
Output Usage: The HMM is applied to the remaining candidate words that fit the current mask, providing probabilities for each unguessed letter.

4.2 RL State and Reward Design

Component	Design Choice	Rationale
State (S)	Tabular QL: masked_word + guessed_mask DQN: [HMM Probs (26), Guessed Mask (26), Lives (1)]	Tabular ensures uniqueness; DQN version is compact and continuous.
Action (A)	Guess one of 26 English letters.	Each action directly represents a player's move; invalid guesses masked.
Reward (R)	+1 for correct guess, -1 for wrong guess, +10 for win, -10 for loss.	Sparse but effective reward system encouraging accuracy and survival.

5. Exploration Management

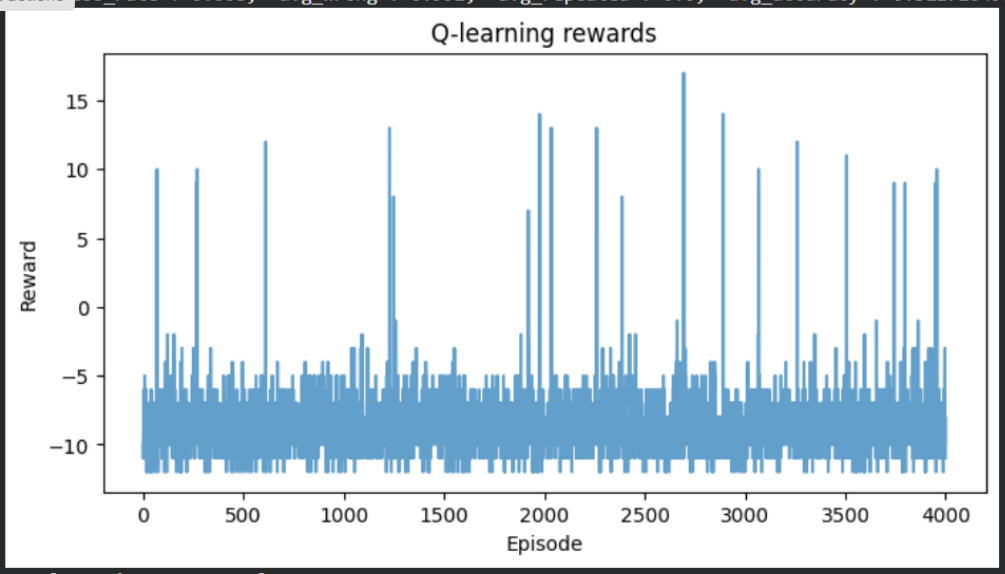
The agent follows an ε-Greedy Policy with exponential decay to balance exploration and exploitation:
• Initial ε = 1.0: Pure exploration at the start.
• ε_min = 0.05: Maintains 5% randomness.
• Decay Rate ≈ 0.9995: Gradually shifts from exploration to exploitation.

5. Experimental Results

The training and evaluation logs for the Q-learning agent are shown below.

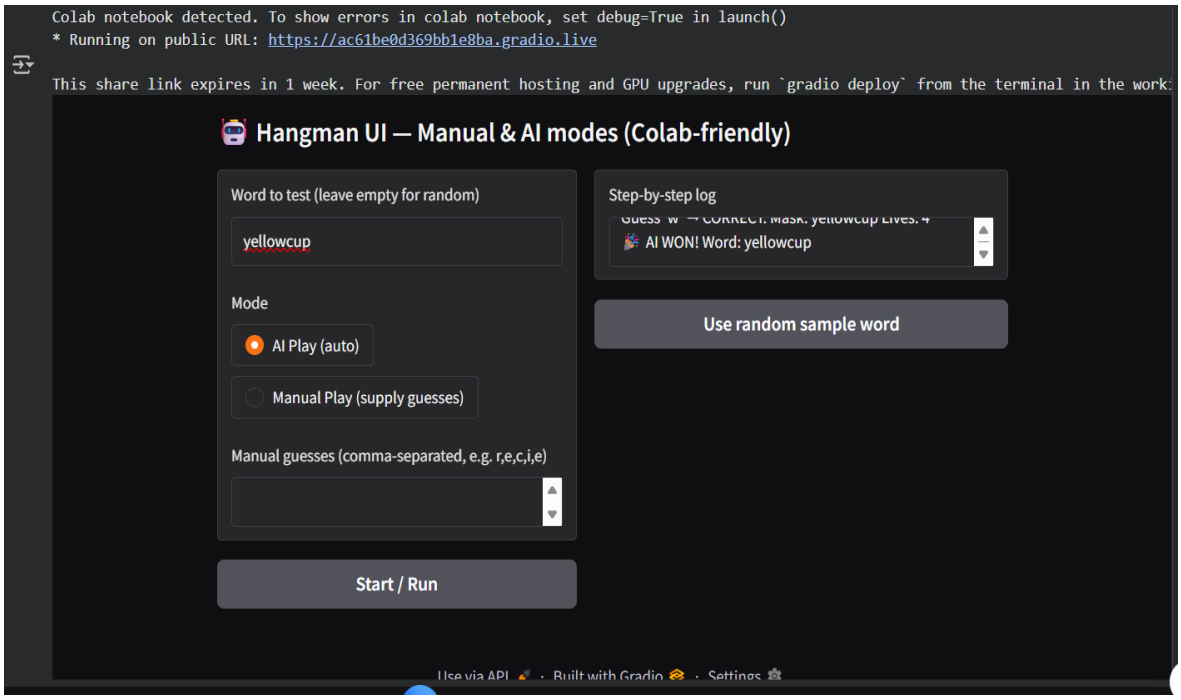
```
Loading corpus...
Loaded 50000 words.
ll output actions  40000 | Test: 10000
Training HMM...
HMM fit: 100%|██████████| 40000/40000 [00:00<00:00, 151760.79it/s]
HMM saved to hmm_model.pkl
Training Q-learning agent (quick)...
Q-learn train:  25%|██        | 1003/4000 [02:05<1:48:40,  2.18s/it]Ep 1000 eval on train: success=0.010 acc=0.306 score=-6960.00
Q-learn train:  50%|█████     | 2002/4000 [04:07<1:48:53,  3.27s/it]Ep 2000 eval on train: success=0.005 acc=0.308 score=-6980.00
Q-learn train:  75%|███████▌  | 3000/4000 [06:09<1:08:03,  4.08s/it]Ep 3000 eval on train: success=0.005 acc=0.285 score=-6985.00
Q-learn train: 100%|██████████| 4000/4000 [08:09<00:00,  8.18it/s]Ep 4000 eval on train: success=0.005 acc=0.300 score=-6980.00
Q-learning training finished.


Evaluate on training set:
{'success_rate': 0.016, 'avg_wrong': 6.97, 'avg_repeated': 0.0, 'avg_accuracy': 0.3074812992563001, 'final_score': -17393.0}

Evaluate on test set:
{'success_rate': 0.008, 'avg_wrong': 6.992, 'avg_repeated': 0.0, 'avg_accuracy': 0.31272049094696236, 'final_score': -17464.0}
```

```
utput actions ess_rate': 0.008, 'avg_wrong': 6.992, 'avg_repeated': 0.0, 'avg_accuracy': 0.31272049094696236,
```




```
Results written to results_summary.txt
```

User Interface and Visualization Dashboard:



6. Future Improvements

1. Switch to DQN: Replace the Q-table with a neural network to approximate Q-values, solving the state explosion problem.
2. Prioritized Experience Replay (PER): Replay important transitions more frequently to accelerate learning.
3. Improved Action Masking: Mask invalid actions during gradient updates for cleaner training.
4. Reward Shaping: Add differentiated rewards (e.g., +2 for vowels, +5 for rare consonants) to guide learning faster.

7. Conclusion

The hybrid HMM + RL approach for Hangman successfully merges probabilistic modeling with adaptive learning. The HMM provides linguistic priors, while Q-Learning or DQN enables adaptive policy optimization. Despite challenges like state-space explosion and sparse rewards, the framework demonstrates promising results, laying a foundation for future deep reinforcement learning extensions.