MACHINE LEARNING HACKTHON

Team Details :

Name	SRN
Parva Shah	PES2UG23CS406
Prajwal Kittali	PES2UG23CS419
Pranav Goankar	PES2UG23CS426
Prashant Radder	PES2UG23CS437

1. Key Observations

The most challenging part of this project was integrating Hidden Markov Models (HMM) and Reinforcement Learning (RL) in a way that balanced linguistic probability with adaptive strategy. The HMM accurately modeled letter frequencies based on the dataset, but it often over-prioritized high-frequency letters like ‘E’ and ‘A,’ which sometimes led to incorrect guesses early on. Designing a dynamic reward mechanism for the RL agent was another tough task — ensuring it learned from mistakes without over-penalizing exploration required fine-tuning parameters such as learning rate, discount factor, and epsilon decay.
One major insight was realizing that pure probabilistic reasoning (HMM) alone couldn’t achieve optimal performance — the agent needed a feedback-driven learning component (RL) to adapt and refine its guessing strategy across games. The combination of the two methods made the system robust and data-efficient.

2. Strategies

The Hidden Markov Model (HMM) was used to estimate the likelihood of each letter based on word structure and contextual frequency from a predefined corpus. Words were categorized by their lengths, and transition probabilities were calculated to predict the most probable next letters for a given masked word pattern. This allowed the agent to make more contextually relevant guesses instead of random frequency-based ones.

In the Reinforcement Learning setup, each game state was represented as a tuple — (current_mask, guessed_letters, remaining_lives). The action space was defined as the set of all unguessed letters. A reward function was crafted to reinforce intelligent behavior:

- +15 points for a correct guess
- +100 for successfully completing the word
- −30 for an incorrect guess
- −100 for losing the game

This reward design encouraged the agent to aim for long-term success rather than short-term random guessing. The Q-learning algorithm used a gradually decaying exploration rate and updated its Q-table after every episode, improving decision-making over time.

3. Exploration vs. Exploitation

To handle the exploration-exploitation trade-off, an epsilon-greedy strategy was employed. Initially, epsilon (ε) was set to 1.0, meaning the agent explored freely, trying various letters even with low probabilities. Over multiple training episodes, epsilon decayed exponentially (e.g., multiplied by 0.9995 per round) until reaching a minimum threshold of 0.01.
This ensured that the agent started by exploring a wide range of letter choices but eventually exploited its learned Q-values to make smarter, data-driven guesses.
Additionally, the final decision-making combined 90% probability weighting from the HMM and 10% learned Q-values from RL, maintaining a balance between statistical reasoning and experiential learning.

4. Future Improvements

If given another week to enhance this project, several improvements could be implemented. First, replacing the Q-table with a Deep Q-Network (DQN) would allow generalization across unseen word patterns, reducing overfitting to the training corpus. Second, adding positional letter probabilities — for example, the likelihood of specific letters appearing in certain positions — could improve contextual guessing accuracy.

Another refinement would be to use dynamic reward shaping, giving higher penalties for repeating letters or missing high-probability guesses. Additionally, introducing transfer learning or Transformer-based models (like BERT embeddings) could incorporate semantic understanding, enabling the system to infer meaning and context from words instead of relying purely on statistics. Lastly, performance could be visualized more thoroughly by plotting the win rate, average reward per episode, and epsilon decay curve, providing deeper insights into the model's learning behavior.