ML HACKATHON – HANGMAN PROJECT

Team 9

1. Key Observations

Q: What were the most challenging parts? What insights did you gain?

A:
The most challenging part was making the Reinforcement Learning (RL) agent understand word patterns with very limited information.
Initially, the agent guessed randomly and often failed to learn any useful pattern.
After integrating the Hidden Markov Model (HMM), the agent started receiving helpful probability hints for likely letters, improving its learning process.

Another challenge was in **reward shaping** — deciding how much to reward correct guesses and penalize wrong ones.
I learned that smaller, consistent rewards for progress work better than large, irregular ones.
Through this project, I gained insight into how **probability models (HMM)** and **learning-based models (DQN)** can work together to improve decision-making.

2. Strategies

Q: What were your HMM design choices? How did you design the RL state and reward system?

A:

HMM Design:

The HMM was used to estimate the probability of each letter being part of the word.

- Hidden states represent the possible positions of letters.
- Emission probabilities were tuned using word patterns.
- The model outputs probabilities for each alphabet, which were given to the RL agent.

RL (DQN) Design:

The RL agent used a **Deep Q-Network (DQN)** to learn which letter to guess next.
The **state representation** included:

- Masked word (e.g., `_ A _ E`)
- Guessed letters so far
- HMM probability distribution for all 26 letters

The **reward structure** was:

- +10 → Correct guess
- –5 → Wrong guess
- –2 → Repeated guess
- +20 → Complete word guessed correctly

This balance of rewards helped the agent reach an **82% win rate** after 2000 episodes.

3. Exploration

Q: How did you manage the exploration vs. exploitation trade-off?

A:
We used the **epsilon-greedy strategy** for exploration.
At the start (ε = 1.0), the agent explored randomly to gather experience.
As training continued, ε slowly decreased to 0.01, so the agent began relying on what it had learned instead of guessing blindly.

Additionally, we used:

- **Prioritized Experience Replay** – important past experiences were replayed more often.
- **Learning Rate Scheduler** – gradually reduced learning rate to stabilize training.

This helped balance **exploration (trying new letters)** and **exploitation (using learned patterns)** efficiently.

4. Future Improvements

Q: If you had one more week, what improvements would you make?

A:
If I had more time, I would:

1. Replace HMM with a **Transformer-based model** for better letter prediction.
2. Add **Double or Dueling DQN** to further stabilize training.
3. Train on **a larger dataset** with longer and more complex words.
4. Introduce **curriculum learning**, starting with easy words and moving to harder ones.
5. Build a **visual dashboard** to display training metrics like rewards, loss, and win rate over time.

These changes would make the agent more adaptive and generalizable.

5. Final Results

Q: What was your final performance score and observation?

A:

- **Episodes Trained:** 2000
- **Best Win Rate:** 82%
- **Average Reward:** +4.58
- **Average Wrong Guesses:** 7.15
- **Final Score: 1600 (Approx.)**

The model successfully exceeded the target accuracy of 70%.
By combining **HMM (for prediction)** and **DQN (for decision-making)**, the agent achieved strong performance and learned efficiently to guess hidden words.

6. Conclusion

The Hangman project showed how **traditional probabilistic models (HMM)** and **modern reinforcement learning (DQN)** can complement each other.
The HMM provided logical probability-based hints, while the DQN learned through rewards and experience.
Together, they formed a smart agent capable of reasoning and adapting like a human player.