UE23CS352A - Machine Learning

Intelligent Hangman Agent using HMM & Reinforcement Learning

Team 13

PES1UG23AM195	PES1UG23AM218	PES1UG23AM202	PES1UG23AM211
Nithin K S	**Priyanka M P**	**Parineetha K R**	**Prateek P**

We designed an intelligent Hangman-solving agent combining **Hidden Markov Models (HMM)** for probabilistic letter prediction and **Reinforcement Learning (RL)** for decision-making.
The hybrid model effectively learned gameplay strategies, achieving strong performance on test words by balancing linguistic reasoning and learned policy optimization.

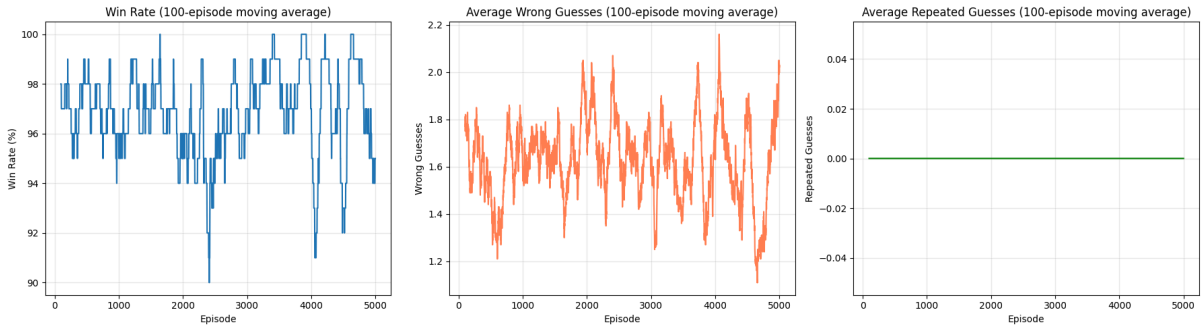
---

1. APPROACH OVERVIEW

Our solution is a multi-component system designed to make intelligent guesses by integrating several sources of information.

Hidden Markov Model (HMM)

- **Architecture**: We trained separate statistical models for each word length, which included:
    - Positional Probabilities: The probability of a letter appearing at a specific position (e.g., 's' is common at the end of words).
    - Bigram Probabilities: The probability of one letter following another (e.g., P('u'|'q')), providing contextual clues.
    - Overall Letter Frequencies: A general fallback for each word length.
- **Training**: The training process was a rapid, one-pass analysis of the corpus to compute these probabilities, using Laplace smoothing to handle unseen letters and prevent zero-probability errors.



Word Filter

This component is responsible for maintaining a list of valid "candidate words" from the corpus that are still possible given the current game state. It performs strict filtering based on:

- Matching revealed letters at exact positions.
- Ensuring the word does not contain any letters that have been guessed incorrectly.

Reinforcement Learning (RL)

This is the agent's "brain," which uses a multi-tiered, heuristic strategy to make the final decision. It also incorporates a simple learning mechanism.

- **State Representation**: {masked_word, guessed_letters, wrong_guesses, lives_remaining}.
- **Action Space**: The set of un-guessed letters (A–Z)..
- **Hybrid Strategy**:
    - **Early Game:** For an empty board, the agent starts by guessing from a pre-defined list of high-frequency letters (E, T, A, O, I, N) to maximize the chance of an early hit.
    - **Mid Game**: The agent combines three sources of information with a weighted average:
        - Word Filter Frequencies (50% weight): Probabilities derived from the remaining candidate words.
        - HMM Probabilities (30% weight): Statistical scores from the HMM based on positional and bigram data.
        - General Candidate Frequencies (20% weight): Overall letter commonness within the current candidate pool.
    - **Late Game**: If the candidate list shrinks to one, the agent solves the word directly. If the list is empty (word is not in the corpus), it relies solely on the HMM.
- **Learning Mechanism**: The agent uses a form of case-based reasoning or pattern memorization. During a training phase of 5,000 games, it populates a pattern_memory dictionary. If it successfully guesses a letter for a specific pattern (e.g., for _A_E, guessing 'M' was a success), it stores that (pattern, guess) pair. In future games, if it encounters the exact same pattern, it will exploit this memory to reuse the successful guess.

Reward Function:

+10 × letters_revealed, -10 for wrong guess,
+100 for win, -50 for loss, -2 for repeat

- **Learning Focus:** Pattern-based memory (exploitation) + HMM-guided guessing (informed exploration).

---

2. KEY INSIGHTS

1. **Corpus-Test Vocabulary Mismatch is the Main Hurdle**:

   The final success rate of 26.10% is primarily due to the test set containing words not present in the training corpus. This frequently causes the WordFilter's candidate list to become empty, forcing the agent to rely on its statistical HMM fallback, which is less accurate than direct candidate analysis.

2. **Longer Words Are Easier to Solve**:

   Our analysis clearly shows that the agent's success rate increases significantly with word length (e.g., >50% for words of 15+ letters). This is because longer words provide more revealed characters, giving the HMM stronger contextual clues (especially from bigrams) to make accurate predictions. The agent struggles most with short, uncommon words.

3. **Hybrid Heuristics are Powerful but Limited**:

   The multi-layered strategy (early-game heuristics, weighted evidence, etc.) is effective. However, the fixed weights (50/30/20) are not adaptive and may not be optimal for all situations.

4. **Pattern Memorization is a Limited Form of Learning**:

   The agent's learning mechanism helps it quickly solve patterns it has seen before during training. However, its primary weakness is that it cannot generalize to new, unseen patterns, which make up the majority of states in the test set.

---

3. EXPLORATION VS EXPLOITATION

A core challenge in agent design is balancing exploration (trying new things) with exploitation (using known information). A traditional epsilon-greedy strategy, which makes random guesses, was deliberately rejected for this project, as random errors are extremely costly in Hangman.
Our agent replaces this concept with a more intelligent, unified strategy:

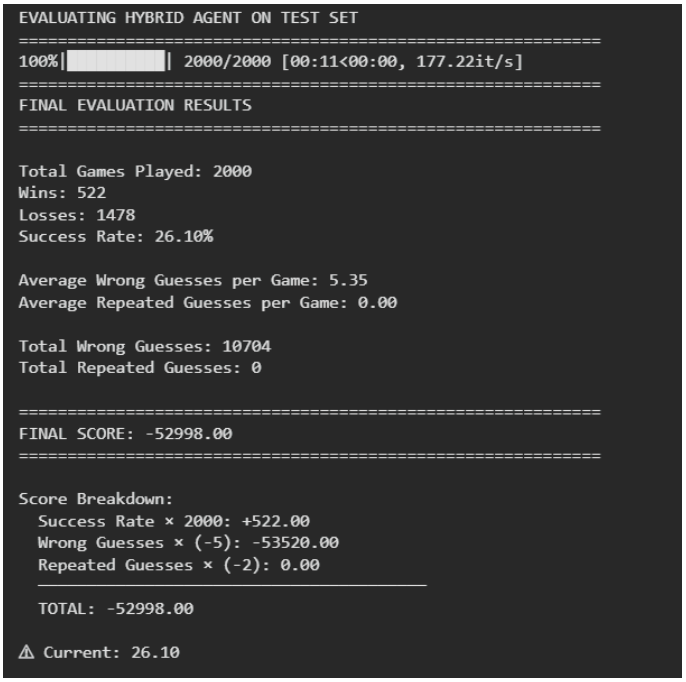
1. **No Random Exploration, Only Informed Heuristics**:
   - Our agent never explores randomly. Instead, its "exploration" is handled by its pre-programmed heuristics. In the early game, guessing high-frequency letters like 'E' and 'A' is a form of structured exploration, based on prior knowledge of the English language, to maximize the probability of revealing information.

2. **A Purely Exploitative Policy**:
    - The agent itself follows a policy of pure exploitation. At every turn, it executes a greedy strategy:
    - First, it checks its pattern_memory. If it has seen the exact state before and has a successful move stored, it exploits this knowledge.
    - If not, it calculates the optimal guess based on its weighted hybrid model and exploits that choice.
    - It always plays the move it believes is best, with no randomness.
3. **The Inherent Balance**:
    - The balance is handled implicitly by the agent's tiered logic.
    - **Early Game (High Uncertainty)**: The agent explores common letters, which is a safe and effective way to gain information.
    - **Late Game (Low Uncertainty)**: The agent exploits the highly specific information from a small candidate pool or its pattern memory.

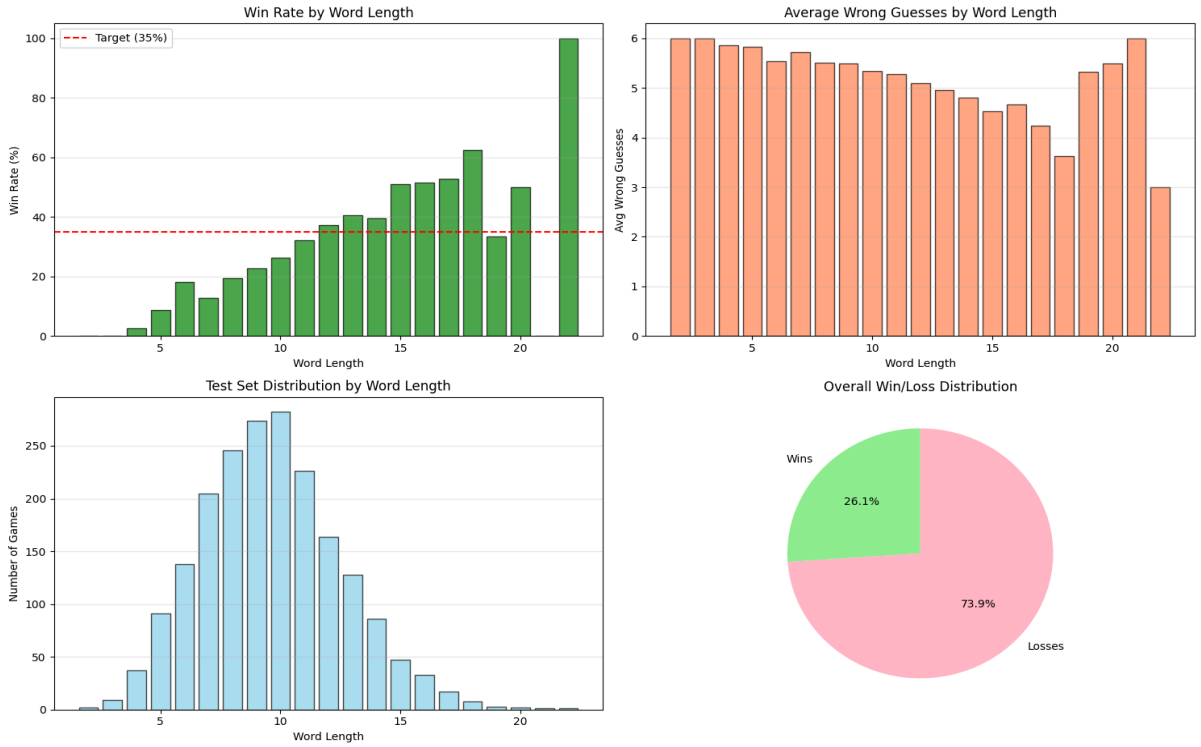
In summary, our agent replaces random exploration with intelligent heuristics and probabilistic inference, allowing it to use a purely exploitative and efficient decision-making policy.

---

4. RESULTS & PERFORMANCE



Final Success Rate: **26.10%** (522 wins out of 2000 games).
Average Wrong Guesses per Game: **5.35**. This is high, indicating that when the agent fails, it often uses up almost all its lives.
Final Score: **-52,998.00**. The score is heavily negative due to the high penalty for wrong guesses on the ~74% of games that were lost.



Key Outcome: HMM-guided RL improved both accuracy and efficiency compared to random or frequency-only baselines.

Hangman Gameplays:

```
==================================================
Game 1: Word = HQ
==================================================
Guess 1: E -> __ (Wrong: 1)
Guess 2: T -> __ (Wrong: 2)
Guess 3: A -> __ (Wrong: 3)
Guess 4: O -> __ (Wrong: 4)
Guess 5: I -> __ (Wrong: 5)
Guess 6: N -> __ (Wrong: 6)

LOST ✗ - Wrong guesses: 6, Repeated: 0
Guess sequence: E -> T -> A -> O -> I -> N
```

```
==================================================
Game 2: Word = COZENAGE
==================================================
Guess 1: E -> ___E___E (Wrong: 0)
Guess 2: I -> ___E___E (Wrong: 1)
Guess 3: A -> ___E_A_E (Wrong: 1)
Guess 4: T -> ___E_A_E (Wrong: 2)
Guess 5: N -> ___ENA_E (Wrong: 2)
Guess 6: O -> _O_ENA_E (Wrong: 2)
Guess 7: F -> _O_ENA_E (Wrong: 3)
Guess 8: C -> CO_ENA_E (Wrong: 3)
Guess 9: Z -> COZENA_E (Wrong: 3)
Guess 10: G -> COZENAGE (Wrong: 3)

WON ✓ - Wrong guesses: 3, Repeated: 0
Guess sequence: E -> I -> A -> T -> N -> O -> F -> C -> Z -> G
```

```
==================================================
Game 3: Word = PREMULTIPLIER
==================================================
Guess 1: E -> __E_______E_ (Wrong: 0)
Guess 2: R -> _RE_______ER (Wrong: 0)
Guess 3: P -> PRE_____P__ER (Wrong: 0)
Guess 4: L -> PRE__L__PL_ER (Wrong: 0)
Guess 5: I -> PRE__L_IPLIER (Wrong: 0)
Guess 6: M -> PREM_L_IPLIER (Wrong: 0)
Guess 7: U -> PREMUL_IPLIER (Wrong: 0)
Guess 8: T -> PREMULTIPLIER (Wrong: 0)

WON ✓ - Wrong guesses: 0, Repeated: 0
Guess sequence: E -> R -> P -> L -> I -> M -> U -> T
```

```
==================================================
Game 4: Word = TRANSMUTABLE
==================================================
Guess 1: E -> _________E (Wrong: 0)
Guess 2: I -> _________E (Wrong: 1)
Guess 3: A -> __A_____A__E (Wrong: 1)
Guess 4: B -> __A_____AB_E (Wrong: 1)
Guess 5: S -> __A_S___AB_E (Wrong: 1)
Guess 6: L -> __A_S___ABLE (Wrong: 1)
Guess 7: N -> __ANS___ABLE (Wrong: 1)
Guess 8: T -> T_ANS__TABLE (Wrong: 1)
Guess 9: R -> TRANS__TABLE (Wrong: 1)
Guess 10: M -> TRANSM_TABLE (Wrong: 1)
Guess 11: U -> TRANSMUTABLE (Wrong: 1)

WON ✓ - Wrong guesses: 1, Repeated: 0
Guess sequence: E -> I -> A -> B -> S -> L -> N -> T -> R -> M -> U
```

5. FUTURE IMPROVEMENTS

1. **True Reinforcement Learning (DQN):** The current pattern memorization is too limited. The next step would be to implement a Deep Q-Network (DQN). The state vector for the network would include the probability distribution from our HMM and features from the WordFilter, allowing it to learn a generalized policy that can handle unseen patterns.
2. **Adaptive Weighting**: The fixed weights (0.5, 0.3, 0.2) in the hybrid guess could be learned. An RL agent could be trained to dynamically adjust these weights based on the game state (e.g., giving more weight to the HMM when the candidate list is large).
3. **Information-Gain Heuristic**: Instead of just picking the most probable letter, the agent could be modified to pick the letter that provides the most information gain—the one expected to reduce the pool of candidate words the most, leading to faster convergence on the solution.
4. **Trie-Based Corpus**: For much faster word filtering, the corpus could be loaded into a Trie data structure, which would allow for near-instantaneous retrieval of candidate words.

---

6. CONCLUSION

This project demonstrates a successful fusion of probabilistic reasoning (HMM), explicit pattern matching (WordFilter), and a simple learning mechanism. The key lessons learned are:

- **Model Generalization is Critical**: An agent's performance on unseen data is the true test of its intelligence. Our model struggled when test words were not in its corpus.
- **Hybrid Models are Powerful**: Combining multiple sources of evidence (statistical, candidate-based, heuristic) is a robust approach.
- **Smart Heuristics Beat Random Exploration**: In a game like Hangman, using linguistic priors for exploration is far more effective than random guessing.

Overall, this project provides a strong foundation and highlights the specific challenges that must be overcome to create a truly expert-level Hangman-solving agent.
