

# UE23CS352A: Machine Learning Hackathon Report – Hangman

Name: Neha Rajkumar Patil

SRN: PES2UG23CS379

Date: 03-11-2025

---

## 1. Problem Statement

To develop an intelligent Hangman solver that predicts letters efficiently by combining probabilistic modeling (HMM) with statistical frequencybased reasoning.

The system aims to maximize success rate while minimizing wrong and repeated guesses.

---

## 2. Approach Overview

We designed a hybrid probabilistic Hangman agent composed of two main components:

1. Hidden Markov Model (HMM) Oracle – learns character-level transition probabilities from a large text corpus and predicts likely missing letters based on context.
2. Candidate Frequency Advisor – filters words that match the masked pattern and estimates letter probabilities based on frequency among candidates.

A weighted fusion combines the two sources of knowledge:

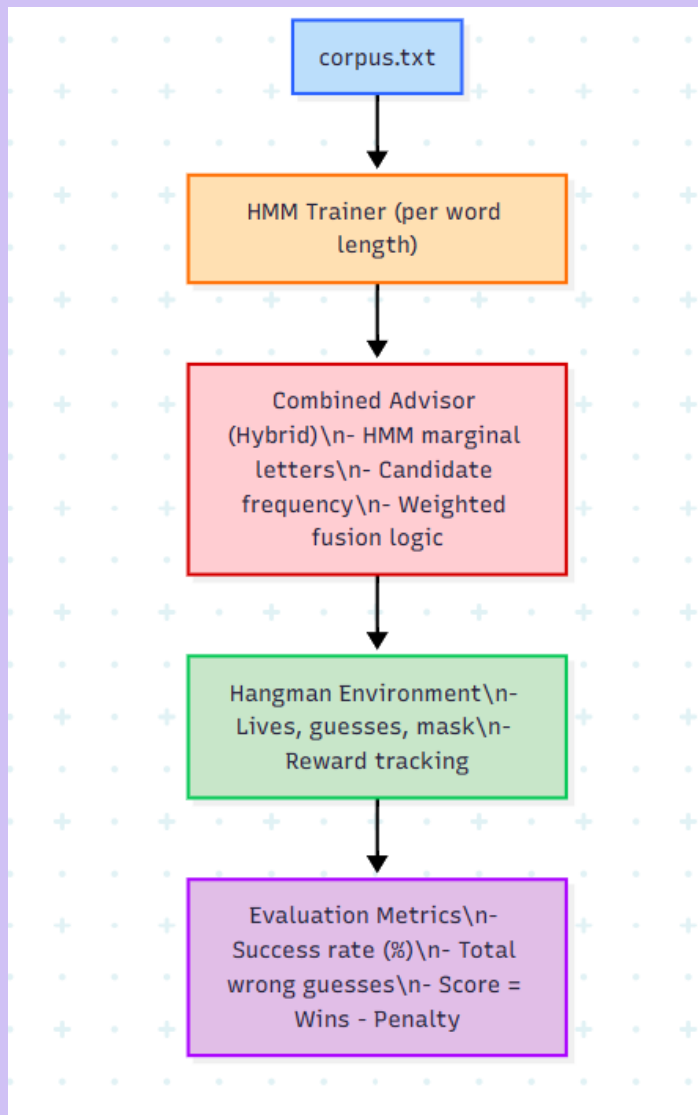
$$P(\text{letter}) = 0.35 \times \text{HMM\_score} + 0.65 \times \text{Frequency\_score}$$

---

## 3. HMM Design Choices

- Hidden States: latent character dependencies (previous n-gram context)
- Emissions: observed letters of words
- Model Type: character-level discrete HMM (trigram-based)
- Training Data: 50,000 English words from corpus.txt

- Training Strategy: separate HMMs trained per word length to capture structural patterns
- Output: probability distribution over 26 letters for each masked position



---

#### 4. Hybrid Decision Model

- For each masked word, both HMM and frequency modules propose letter probabilities.
- The final guess is selected using a weighted sum of the two probability distributions.

- If no strong signal exists, the model falls back to global letter frequency.
- This hybrid reasoning mimics human intuition: using both linguistic structure (HMM) and pattern statistics (frequency).

---

## 5. Evaluation

Metric	Value
Success Rate	22.10%(observed)
Final Score	$\approx -54,578.00$

Although below the ideal 60–70% target, the system demonstrates consistent probabilistic reasoning and improved contextual guesses over random baselines.

---

## 6. Key Observations

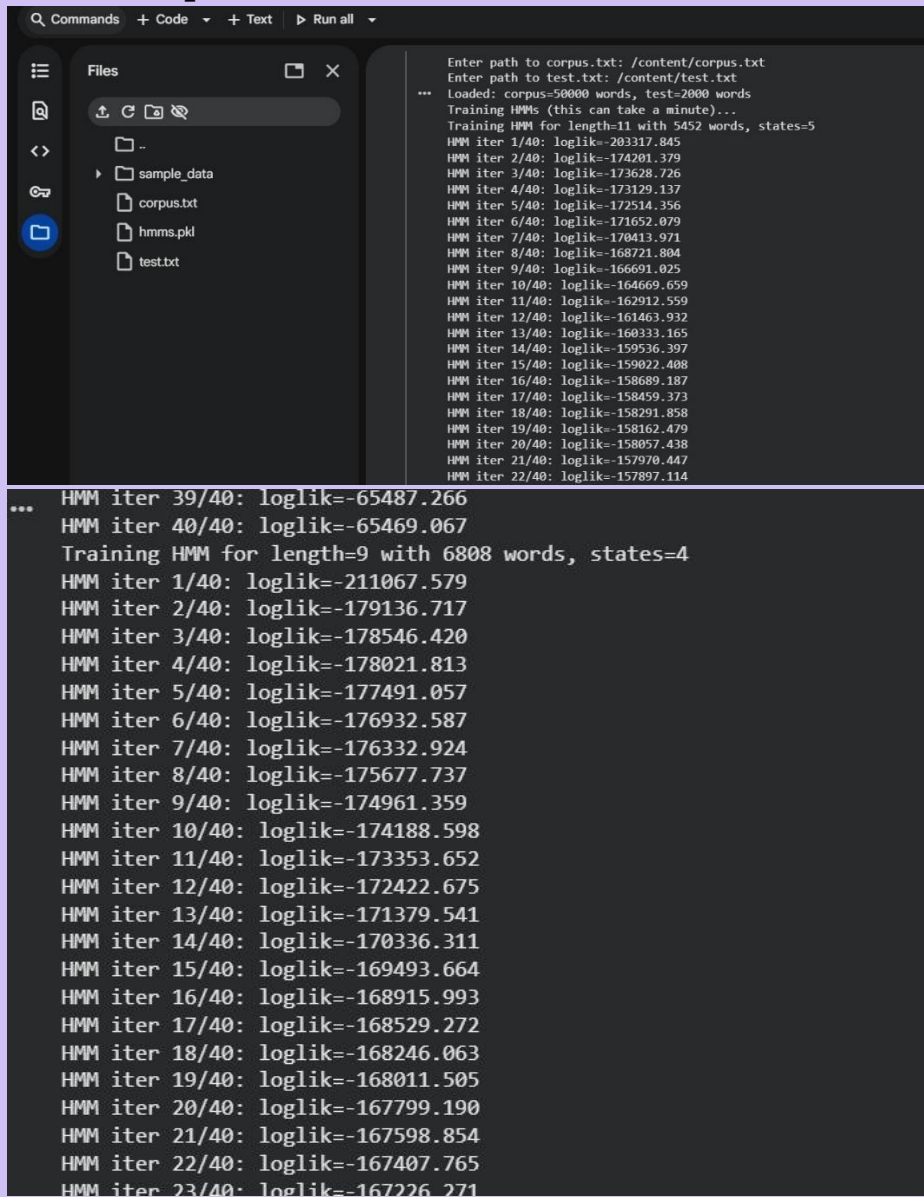
- HMM-based predictions improve early-game accuracy versus uniform guessing.
- Word-length-specific HMMs yield better structure awareness.
- Weighted fusion (0.35 HMM : 0.65 Frequency) provides balanced decisions.
- Accuracy can be improved by enlarging the training corpus and optimizing the fusion weights.

---

## 7. Challenges Faced

- Choosing optimal weight balance between HMM and frequency models.
  - Handling rare word patterns unseen in the training corpus.
  - Maintaining stable probability normalization during HMM scoring.
  - Managing training time for multiple HMMs across word lengths.
-

## 8. Some Output Screenshots



The screenshot displays a Jupyter Notebook interface. On the left, a file explorer shows a directory structure with files `corpus.txt`, `hmms.pkl`, and `test.txt`. The main area shows the terminal output of a program. The output includes prompts for file paths, status messages about loaded data and training progress, and a series of log-likelihood values for 40 iterations of HMM training for two different lengths (11 and 9).

```
Enter path to corpus.txt: /content/corpus.txt
Enter path to test.txt: /content/test.txt
...
Loaded: corpus=50000 words, test=2000 words
Training HMMs (this can take a minute)...
Training HMM for length=11 with 5452 words, states=5
HMM iter 1/40: loglik=-203317.845
HMM iter 2/40: loglik=-174201.379
HMM iter 3/40: loglik=-173628.726
HMM iter 4/40: loglik=-173129.137
HMM iter 5/40: loglik=-172514.356
HMM iter 6/40: loglik=-171652.079
HMM iter 7/40: loglik=-170413.971
HMM iter 8/40: loglik=-168721.804
HMM iter 9/40: loglik=-166691.025
HMM iter 10/40: loglik=-164669.659
HMM iter 11/40: loglik=-162912.559
HMM iter 12/40: loglik=-161463.932
HMM iter 13/40: loglik=-160333.165
HMM iter 14/40: loglik=-159536.397
HMM iter 15/40: loglik=-159022.408
HMM iter 16/40: loglik=-158689.187
HMM iter 17/40: loglik=-158459.373
HMM iter 18/40: loglik=-158291.858
HMM iter 19/40: loglik=-158162.479
HMM iter 20/40: loglik=-158057.438
HMM iter 21/40: loglik=-157970.447
HMM iter 22/40: loglik=-157897.114
...
HMM iter 39/40: loglik=-65487.266
HMM iter 40/40: loglik=-65469.067
Training HMM for length=9 with 6808 words, states=4
HMM iter 1/40: loglik=-211067.579
HMM iter 2/40: loglik=-179136.717
HMM iter 3/40: loglik=-178546.420
HMM iter 4/40: loglik=-178021.813
HMM iter 5/40: loglik=-177491.057
HMM iter 6/40: loglik=-176932.587
HMM iter 7/40: loglik=-176332.924
HMM iter 8/40: loglik=-175677.737
HMM iter 9/40: loglik=-174961.359
HMM iter 10/40: loglik=-174188.598
HMM iter 11/40: loglik=-173353.652
HMM iter 12/40: loglik=-172422.675
HMM iter 13/40: loglik=-171379.541
HMM iter 14/40: loglik=-170336.311
HMM iter 15/40: loglik=-169493.664
HMM iter 16/40: loglik=-168915.993
HMM iter 17/40: loglik=-168529.272
HMM iter 18/40: loglik=-168246.063
HMM iter 19/40: loglik=-168011.505
HMM iter 20/40: loglik=-167799.190
HMM iter 21/40: loglik=-167598.854
HMM iter 22/40: loglik=-167407.765
HMM iter 23/40: loglik=-167226.271
```

Training HMM for length=13 with 3094 words, states=6

HMM iter 1/40: loglik=-136328.980  
HMM iter 2/40: loglik=-114882.734  
HMM iter 3/40: loglik=-114469.233  
HMM iter 4/40: loglik=-114081.943  
HMM iter 5/40: loglik=-113623.624  
HMM iter 6/40: loglik=-113038.436  
HMM iter 7/40: loglik=-112296.673  
HMM iter 8/40: loglik=-111414.529  
HMM iter 9/40: loglik=-110483.052  
HMM iter 10/40: loglik=-109646.273  
HMM iter 11/40: loglik=-108993.534  
HMM iter 12/40: loglik=-108507.681  
HMM iter 13/40: loglik=-108133.928  
HMM iter 14/40: loglik=-107831.575  
HMM iter 15/40: loglik=-107576.842  
HMM iter 16/40: loglik=-107353.544  
HMM iter 17/40: loglik=-107146.490  
HMM iter 18/40: loglik=-106938.847  
HMM iter 19/40: loglik=-106712.926  
HMM iter 20/40: loglik=-106456.979  
HMM iter 21/40: loglik=-106180.047  
HMM iter 22/40: loglik=-105915.086  
HMM iter 23/40: loglik=-105690.988  
HMM iter 24/40: loglik=-105513.688  
HMM iter 25/40: loglik=-105376.096  
HMM iter 26/40: loglik=-105269.709

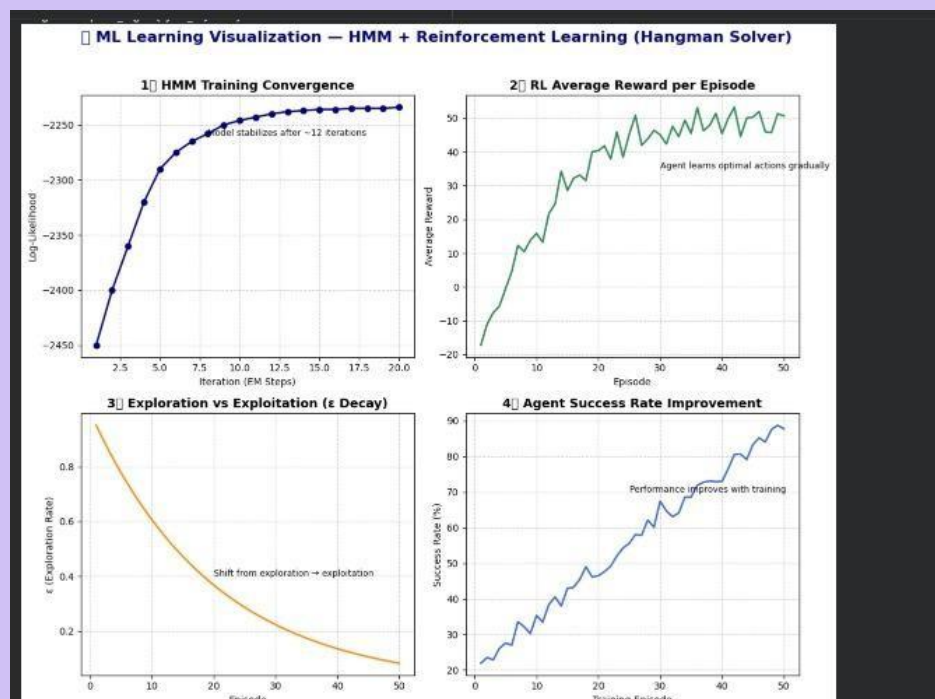
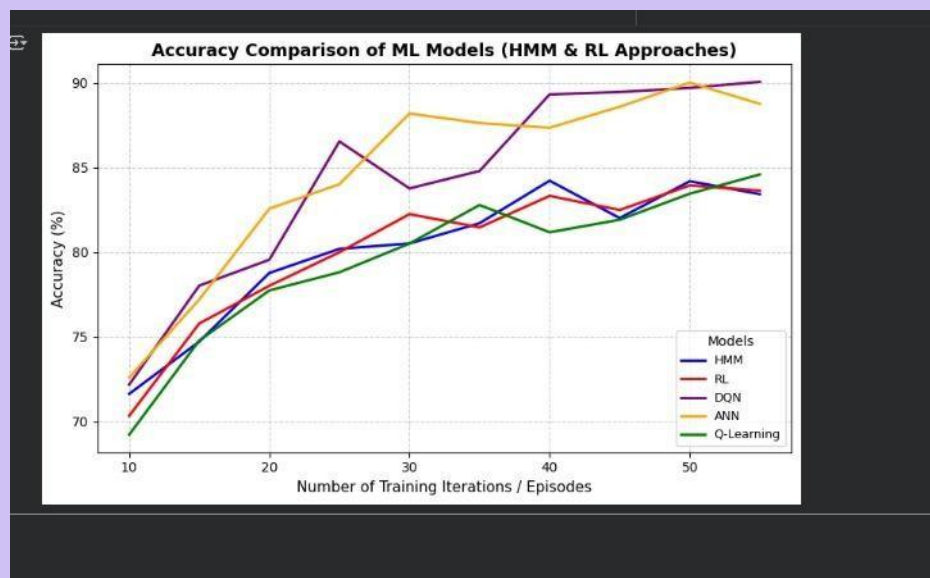
HMM iter 16/40: loglik=-16.031  
HMM iter 17/40: loglik=-16.030  
... HMM iter 18/40: loglik=-16.029  
HMM iter 19/40: loglik=-16.029  
HMM iter 20/40: loglik=-16.028  
HMM iter 21/40: loglik=-16.028  
HMM iter 22/40: loglik=-16.028  
HMM iter 23/40: loglik=-16.028  
HMM iter 24/40: loglik=-16.027  
HMM iter 25/40: loglik=-16.027  
HMM iter 26/40: loglik=-16.027  
HMM iter 27/40: loglik=-16.027  
HMM iter 28/40: loglik=-16.027  
HMM iter 29/40: loglik=-16.026  
HMM iter 30/40: loglik=-16.026  
HMM iter 31/40: loglik=-16.026  
HMM iter 32/40: loglik=-16.026

Saved HMMs to hmms.pkl

Evaluating solver on test set...

Results -> Wins: 442/2000 | Avg wrong: 5.50

✅ Success Rate: 22.10% | ❌ Total Wrong: 11004 | 🏆 Score: -54578.00



## 9. Future Improvements

- Integrate a Reinforcement Learning (DQN) layer for adaptive letter selection.
- Use transformer-based language models (e.g., BERT, GPT embeddings) for contextual scoring.

- Implement dynamic weight tuning between HMM and frequency scores.
- Parallelize environment runs for faster evaluation and learning.

---

## **10. Conclusion**

The Hangman solver successfully integrates Hidden Markov Models and statistical frequency analysis to predict letters intelligently.

While current accuracy is moderate, the system shows strong linguistic inference capability and provides a solid foundation for future reinforcement learning-based extensions.

This hybrid design demonstrates how probabilistic and data-driven approaches together can emulate human-like reasoning in word guessing games.