Analysis Report

Team 1

- Amrutha Hegde: PES1UG23CS064
- Anusha Balamurali Mysore: PES1UG23CS086
- Atharv Rawal: PES1UG23CS119
- Anirudh Sudhir: PES1UG23CS917

1. Key Observations and Insights

Dataset Characteristics

- **Training Corpus**: 50,000 words with diverse lengths (1-24 characters)
- **Test Set**: 2,000 words with zero overlap with training corpus
- **Critical Insight**: Zero overlap necessitates pattern learning over word memorization

Performance Analysis

- **Final Score**: -51,288
- **Success Rate**: 32.85% (657/2000 games won)
- **Average Wrong Guesses**: 5.19 per game
- **Average Repeated Guesses**: 0.0 (perfect efficiency)

Scoring Formula Impact

```
Score = (Success Rate × 2000) - (Total Wrong × 5) - (Total Repeated × 2)
Score = (657) - (10,389 × 5) - (0 × 2)
Score = 657 - 51,945 = -51,288
```

The 5× penalty for wrong guesses heavily dominates the scoring, requiring >50% success rate for positive scores.

Pattern Discovery

1. Position-based letter frequencies vary significantly by word length
2. N-gram dependencies provide strong predictive signals
3. Trigrams offer best context when available (known letters)
4. Bigrams effective for partial patterns
5. Unigrams serve as fallback for minimal information states

2. HMM Strategy and Design

Architecture Overview

The Hidden Markov Model employs:

- 26 hidden states (one per letter a-z)
- Position-dependent emissions trained separately for each word length (1-24)
- Forward-Backward algorithm for probabilistic inference

Training Process

Emission Probability Calculation For each word length L and position p:

$$P(\text{letter}_i|\text{position}_p, \text{length}_L) = \frac{\text{count}(\text{letter}_i, p, L)}{\text{total letters at position } p \text{ for length } L}$$

Forward-Backward Algorithm Given observed pattern O (revealed letters and blanks):

Forward Pass:

$$\alpha_t(i) = P(\text{letter}_i|O_{1:t})$$

Backward Pass:

$$\beta_t(i) = P(O_{t+1:T}|\text{letter}_i)$$

Posterior Probability:

$$\gamma_t(i) = \frac{\alpha_t(i) \cdot \beta_t(i)}{\sum_j \alpha_t(j) \cdot \beta_t(j)}$$

N-gram Integration

Unigram Model

$$P(\text{letter}) = \frac{\text{count(letter)}}{\text{total letters in corpus}}$$

Bigram Model

$$P(\text{letter}_2|\text{letter}_1) = \frac{\text{count}(\text{letter}_1, \text{letter}_2)}{\text{count}(\text{letter}_1)}$$

Trigram Model

$$P(\text{letter}_3|\text{letter}_1, \text{letter}_2) = \frac{\text{count}(\text{letter}_1, \text{letter}_2, \text{letter}_3)}{\text{count}(\text{letter}_1, \text{letter}_2)}$$

Laplace Smoothing To handle unseen n-grams:

$$P_{\text{smoothed}} = \frac{\text{count} + \alpha}{\text{total} + \alpha \times |\text{vocabulary}|}$$

Where $\alpha \in [0.1, 0.5]$ based on n-gram order.

Probability Combination Strategy

$$P_{\text{final}}(\text{letter}) = w_1 \cdot P_{\text{HMM}} + w_2 \cdot P_{\text{bigram}} + w_3 \cdot P_{\text{trigram}} + w_4 \cdot P_{\text{unigram}}$$

Optimal Weights (determined empirically):

- $w_1 = 3.0$ (HMM emission probabilities)
- $w_2 = 2.0$ (Bigram context)
- $w_3 = 2.5$ (Trigram context - strongest when available)
- $w_4 = 1.0$ (Unigram baseline)

Why This Approach Works

1. **Position awareness**: Different letters appear at different frequencies by position
2. **Context exploitation**: N-grams capture letter co-occurrence patterns
3. **Graceful degradation**: Falls back to weaker models when context unavailable
4. **Probabilistic framework**: Handles uncertainty inherently through probability distributions

3. RL Agent Design

State Representation

$$S_t = (\text{pattern}_t, \text{guessed\_letters}_t, \text{wrong\_count}_t)$$

Where:

- pattern_t: Current word state (e.g., "_ a _ _ l e")
- guessed_letters_t: Set of previously guessed letters
- wrong_count_t: Number of incorrect guesses (max 6)

Action Space

$$A = \{a, b, c, ..., z\} \setminus \text{guessed\_letters}_t$$

26 possible actions (letters), filtered to remove already-guessed letters.

Reward Function

$$R(s,a) = \begin{cases} +5 & \text{if letter } a \text{ appears in word} \\ -10 & \text{if letter } a \text{ not in word} \\ +100 & \text{if game won} \\ -50 & \text{if game lost} \end{cases}$$

Policy Strategy

Greedy Policy: Always select the letter with highest probability from HMM+N-gram model.

$$\pi(s) = \arg\max_{a \in A} P_{\text{final}}(a|s)$$

Rationale: Deterministic letter probabilities don't benefit from exploration; exploitation of learned patterns is optimal.

Training Dynamics

- **500 training episodes** on corpus words
- **Episodic learning**: Each game is independent
- **No gradient updates**: Pre-trained HMM provides action probabilities
- **Learning curve**: Reward stabilizes around -15 to -25 per episode (reflecting inherent game difficulty)

4. Exploration Management

Why Minimal Exploration Was Chosen

Traditional RL Exploration Typical RL agents use ϵ-greedy or softmax exploration to discover better actions. However, Hangman presents unique characteristics:

1. **Deterministic Probabilities**: Letter probabilities from HMM are fixed after training
2. **No Hidden Rewards**: All game outcomes are immediately observable
3. **Perfect Information**: Training corpus provides complete pattern statistics
4. **No State Aliasing**: Each pattern uniquely determines optimal letter distribution

Exploration Experiments Conducted Forced vowel-first exploration

- Strategy: Always guess vowels (a, e, i, o, u) before consonants
- Result: **-57,820 score (worst performance)**
- Lesson: Ignoring probabilities for forced exploration hurts performance

Q-Learning with $\epsilon = 0.1$

- Strategy: 10% random exploration during training
- Result: **-56,492 score**
- Problem: State space too large (pattern × guessed combinations)
- Lesson: Exploration doesn't help when optimal policy is pattern-dependent

Pure greedy exploitation

- Strategy: Always select highest probability letter
- Result: **-51,288 score (best performance)**
- Success: Maximizes use of learned statistical patterns

Exploration Trade-offs

Strategy	Pros	Cons	Result
-greedy	Discovers suboptimal patterns	Wastes guesses on low-probability letters	-56,492
Forced vowels	Ensures common letters tried	Ignores word-specific patterns	-57,820
Greedy	Maximizes pattern utilization	No discovery of new strategies	-51,288

Conclusion on Exploration

For Hangman with pre-trained models:

- Pattern probabilities encode optimal strategy
- Random exploration introduces unnecessary errors
- Deterministic greedy policy achieves best performance

5. Future Improvements

Short-term Enhancements

1. Advanced Smoothing Techniques

- **Kneser-Ney smoothing** for better N-gram probability estimates
- **Backoff models** that systematically degrade from trigram → bigram → unigram
- Expected improvement: +2-3% success rate

2. Word Length-Specific Strategies

- Separate models for short words (4 letters) vs long words (10 letters)
- Short words: Prioritize vowels and common letters
- Long words: Leverage stronger N-gram patterns
- Expected improvement: +1-2% success rate

3. Dynamic Weight Adjustment

$$w_i(t) = f(\text{revealed\_ratio}_t, \text{wrong\_count}_t)$$

- Increase HMM weight early (rely on position)
- Increase N-gram weight late (rely on context)
- Expected improvement: +1-2% success rate

Medium-term Enhancements

4. Ensemble Methods Combine multiple models:

- HMM + N-grams (current)
- Neural language model (transformer)
- Phonetic similarity model
- Voting or weighted averaging
- Expected improvement: +3-5% success rate

5. Reinforcement Learning with Function Approximation

- **Deep Q-Network (DQN)** with pattern embeddings
- Neural network to learn Q(pattern, letter)
- Overcomes state space explosion through generalization
- Expected improvement: +2-4% success rate

6. Conclusion

This Hangman AI agent demonstrates that combining Hidden Markov Models with N-gram language models provides strong performance on unseen words. The greedy exploitation strategy outperforms exploration-based approaches due to the deterministic nature of letter probabilities. Future work should focus on more sophisticated language models and ensemble techniques to push beyond the current 32.85% success rate.

Key Takeaways

1. Pattern learning > word memorization for zero-overlap datasets
2. Position-based and context-based probabilities are complementary
3. Greedy policies excel when optimal distributions are learnable
4. N-gram context provides crucial disambiguation signal
5. Proper probability smoothing essential for robustness

Final Performance Summary

- Score: -51,288
- Success Rate: 32.85%
- Wrong Guesses: 5.19 avg
- Repeated Guesses: 0.0 avg
- Best Model: HMM + Trigram/Bigram/Unigram ensemble (V8)