Trexquant Interview Project
(The Hangman Game)

Rohit Karwa

November 26, 2024

1 Introduction

Hangman, a popular word-guessing game, is a compelling challenge due to its linguistic and computational intricacies. As part of the TrexQuant internship selection process, I was tasked with developing an algorithm that outperforms a basic model in playing Hangman. The target was an accuracy rate of over **50%**.

This report outlines the development of my Hangman-playing algorithm, which utilizes linguistic patterns, conditional probabilities, and statistical techniques to improve guessing accuracy.

2 Intuition

Hangman involves guessing a hidden word represented by blanks (_). The game ends when the word is correctly guessed or **6 incorrect guesses** are made.

Key Observations

1. **Frequent Letter Patterns:** Common patterns like *QU*, endings such as *TION*, *MENT*, and *NESS* improve predictive accuracy.

2. **Word Length:** Short words (5–6 letters) have fewer identifiable patterns and are harder.

3. **Dummy Variables:** Special symbols { (start) and — (end) were added to words to capture positional structure such as *ING*, *TION*.

4. **N-grams:** N-grams up to length 5 were used to balance prediction accuracy and computation.

3 Methodology

Step 1: Initialization

Start with the hidden pattern e.g., APPLE ⇒ _ _ _ _ _
After adding dummy variables: { _ _ _ _ _ —

Step 2: Guess Refinement

Example: After guessing letters P and E: { _ P P _ E —

Step 3: Pattern Breakdown

Table 1: N-grams with 1 blank

α_1	α_2	α_3	α_4	α_5
_	{_	{_P	{_PP	PP_E—
_	_P	_PP	PP_E	
	P_	PP_	P_E—	
	_E	P_E		
		_E—		

Table 2: N-grams with 2 blanks

β_1	β_2
_PP_ (β_{11})	{_PP_ (β_{21})
	_PP_E (β_{22})

Step 4: Score Computation

For a letter K,

$$\text{Score}(K) = \sum_{i=1}^{5} C_i \sum_{j=1}^{n_i} P(\gamma_{ij} = K \mid \alpha_{ij}, K \notin guessed) + \sum_{i=1}^{5} C_i \sum_{j=1}^{m_i} [P(\gamma_{ij1} = K) + P(\gamma_{ij2} = K) - P(\gamma_{ij1} = K \wedge$$

Step 5: Conditional Probabilities

Single Blank

$$P(\gamma_{ij} = K) = \frac{N(PKE)\,\mathbb{I}(K \notin guessed)}{\sum_{\delta \notin guessed} N(P\delta E)}$$

Two Blanks

$$P(\gamma_{ij1} = K) = \frac{N(KPP\delta)}{\sum_{\delta_1,\delta_2} N(\delta_1 PP \delta_2)}$$

$$P(\gamma_{ij2} = K) = \frac{N(\delta PPK)}{\sum_{\delta_1,\delta_2} N(\delta_1 PP \delta_2)}$$

$$P(\gamma_{ij1} = K \wedge \gamma_{ij2} = K) = \frac{N(KPPK)}{\sum_{\delta_1,\delta_2} N(\delta_1 PP \delta_2)}$$

Step 6: Letter Selection

Choose the letter with the highest score.

Step 7: Update Pattern

Insert guessed letter and repeat until the word is found or 6 errors occur.

4 Functions

Function	Description
guess(word)	Selects best letter based on combined probabilities.
find_score(word, ngram)	Computes scores of all letters from n-grams.
Conditional_Prob1(pat)	Probability for 1-blank patterns.
Conditional_Prob2(pat)	Probability for 2-blank patterns.
normalize1(pat)	Converts raw frequency to probability (1 blank).
normalize2(pat)	Converts raw frequency to probability (2 blanks).
build_freq_tables()	Precomputes pattern frequency tables.

5 Results

- Achieved **66.5%** word-guessing accuracy.
- Exceeded benchmark performance of 50%.

6 Conclusion

This project demonstrates how linguistic structures combined with statistical modeling can significantly improve Hangman gameplay. By leveraging n-gram components and conditional probabilities, the algorithm outperformed a basic implementation and achieved successful predictive accuracy.