Overview of My Trexquant Interview Project: The Hangman Game

Roy Luo

May 19, 2024

Introduction

Hi my name is Roy Luo and this is an overview of my Trexquant interview project (The Hangman Game). I achieved a success rate of **47%** using **statistical analysis** and **information theory**.

My solution utilizes a combination of strategic letter frequency analysis, dictionary filtering, and entropy calculations for optimal letter selection. This multifaceted approach optimally balances the revealing of new letters and minimizes the risk of incorrect guesses. Here is an in-depth look at each component:

Dictionary Setup and Letter Frequency Analysis

The foundational step in the algorithm is to prepare the word dictionary and analyze the frequency of letters by loading the dictionary of 250,000 words, which serves as the training set.

```
def build_dictionary(file_location):
    with open(file_location, 'r') as file:
        return [line.strip() for line in file]
```

Letter Frequency Computation

The frequency of each letter across all dictionary words is computed to inform the initial guesses. This frequency is calculated by counting each letter in a word only once, thereby focusing on the diversity of letters rather than their total occurrence.

```
def calculate_letter_frequencies(dictionary):
    letter_count = collections.Counter()
    for word in dictionary:
        letter_count.update(set(word))
    return letter_count
```

Game Strategy

Pattern Matching: The algorithm narrows down the list of possible words by matching the revealed pattern to words in the dictionary. It uses regular expressions to filter words that fit the current known pattern of the word.

Pattern Matching
Using regular expressions, the algorithm filters potential words that fit the known pattern of the word.

```
def update_possible_words(pattern):
    regex_pattern = pattern.replace('_', '.')
    self.current_dictionary = [word for word in self.
       ↪ full_dictionary if re.fullmatch(regex_pattern, word)]
```

Entropy-Based Guess Selection
The entropy is used as a measure to choose the most informative next guess. Entropy is calculated for each letter not yet guessed, based on the distribution of words that would result from guessing that letter.

Entropy Calculation
The entropy, $H(X)$, for a random variable X is defined by:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

where $p(x_i)$ is the probability of each configuration when a letter is guessed. In the context of Hangman, entropy is calculated for each potential guess by considering how the set of possible words is divided into subsets based on the presence of the guessed letter. Each subset corresponds to a different configuration of the word that includes the guessed letter at specific positions or excludes it entirely.

```
def calculate_entropy(letter):
    subsets = collections.defaultdict(list)
    for word in self.current_dictionary:
        key = ''.join([char if char == letter else '_' for char
           ↪ in word])
        subsets[key].append(word)
    total = sum(len(words) for words in subsets.values())
    entropy = -sum((len(words) / total) * math.log2(len(words) /
       ↪ total) if len(words) / total > 0 else 0 for words in
       ↪ subsets.values())
    return entropy
```

Game Simulation and Testing
he guess method integrates the entropy calculation to select the optimal letter. This approach balances the likelihood of revealing more letters against the risk of an incorrect guess.

```
def guess(pattern):
    self.update_possible_words(pattern.replace("␣", ""))
    best_guess, max_entropy = None, -float('inf')
    letters = set('abcdefghijklmnopqrstuvwxyz') - self.
       ↪ guessed_letters
    for letter in letters:
```

```
            entropy = self.calculate_entropy(letter)
            if entropy > max_entropy:
                best_guess, max_entropy = letter, entropy
        return best_guess if best_guess else random.choice(list(
            ↪ letters))
```

Conclusion

The entropy-based Hangman algorithm maximizes the informational gain of each guess, enhancing both efficiency and effectiveness. This strategic approach not only exceeds benchmarks but also lays a foundation for further enhancements and theoretical advancements.