An Information Theoretic Formalization of Hangman and Battleship

Bowen Yin

The Harker School

December 11, 2020

Introduction

The classic games of Hangman and Battleship are related in that they both require the player to minimize the number of guesses needed to win the game. With this knowledge, we can formalize these games using information theory.

We would like to maximize the amount of information that can be extracted with each guess. By gathering the most amount of information possible, we can narrow down subsequent guesses and overall lessen the total number of guesses required.

Hangman

In Hangman, the source is simply the player who guesses letters, and in this case, the role is performed by a computer. The receiver is the person who chooses the word at the start of the game and replies to each guess. For the classic version of Hangman, we can assume that there is no noise, since the guess is directly communicated to the receiver and no information is lost. However, for the version where the receiver is allowed to lie up to one time throughout the game, noise is introduced since there is a possibility that the resulting response doesn't match the letter that was guessed. First, we'll discuss a way to minimize the number of guesses in the classic version of Hangman.

To make each guess as useful as possible, we would need to maximize the entropy of each guess. For

each guess, there are two possible outcomes: the letter appears somewhere in the phrase or it doesn't. If we maximize the probability that a letter appears in the word by making it as close to 50% as possible, we can maximize the amount of information that gives us.

An obvious approach would be to guess the letters of the alphabet in descending order by frequency (such as 'e,' 'a,' etc.). However, this may start to be inefficient after a few letters are guessed due to the nature of English spelling. Vowels are much more common than consonants in English, but if a word contains a vowel, that reduces the chance that another vowel exists in the word and increases the chance that there is a consonant. Simply going in order of frequency would result in guessing all of the vowels towards the beginning and leaving the consonants towards the end. Similarly, guessing 'q' essentially guarantees that a 'u' will follow. These and many other specific cases would be difficult to handle without considering hundreds of grammatical rules.

After some thinking, I thought a way to generalize these rules would be to use a list of English words. At the start, we can narrow down the dictionary to only words of the specified length, which may change the relative frequency of letters, After guessing a letter correctly, we can further filter the dictionary to only include words with that letter in those exact positions. For example, if the word is `EXECUTE`, we would query the dictionary for words matching the pattern `E_E___E` after guessing the letter `E`. This significantly reduces the number of possible letters, making subsequent guesses much more useful. For each subset of words that we use from the dictionary, we recompute the letter frequencies by summing up the number of words with each letter, which helps determine the next letter that should be guessed.

For the version of Hangman where the receiver can lie up to one time, we can again filter the dictionary to determine all of the possible words. However, this time we can simulate an incorrect letter by considering letters that have already been guessed before and weighing that case by the probability of the lie occurring. This way, we can account for the possibility of a lie and still take advantage of filtering through the dictionary.

An additional step I took was weighing each word by its frequency in English. The word list is sorted by frequency, so words in the first couple of thousand lines would have a greater weight than words near the bottom of the list, since they are generally more likely to be chosen. This weighing was factored in when calculating the frequency of each letter.

Battleship