Hangman and Battleship

Jerry Chen

Period 1

1 Hangman

1.1 Objective

In hangman, the player must try to find their opponent's phrase in by guessing letters. Although traditionally, the game is over after a limited number of guesses, for the purposes of this project, the goal is simply to identify the phrase in as few guesses as possible. The player-in this case, the computer program-can guess either a single letter or an entire phrase, in which case an incorrect phrase costs the player three guesses.

1.2 Implementation

The program began by reading a given dictionary file that contained 333,333 words that were ordered by their frequencies in the English language. Once the program stored the words, it asked for the length of the phrase. In order to first learn how many words are in the phrase and the length of each word, of course, the initial guess must be 'space.' A subset of the base dictionary was created for each word, filtered on the word length. After splitting the phrase, however, some metric must be applied to determine which letter to guess next.

Naturally, some letters will be more common than others, making them more advantageous to guess. For example, 'E' is far more likely to be contained in the mystery phrase than 'X' or 'Q,' giving it more weight in the consideration of which letter to guess. Thus, a simple method could have been to just count the number of times each letter appears in the available dictionary for the word and guessing the most common letter each time. Hypothetically, however, the least common words in the dictionary could have a plethora of 'Xs' and 'Qs,' and while 'E' might be found in the most common words, it might not appear as many times. Consequently, the rational decision was to weight each letter not only on the number of times it appears but also on the frequency of the word in which it appears.

At this point, the program's strategy made relatively smart guesses, but it could be further optimized. Because the goal was to make the fewest number of guesses possible, if the program was confident enough in the identity of one of the words in the phrase, it would ideally make guesses to reveal the other words. For example, let the phrase be ”INFORMATION THEORY.” After a few guesses-perhaps T, H, O, and Y (note that these are not necessarily the letters the program guessed)-we would know the phrase to be ”___O___T_O_ TH_O_Y.” The next guess might be 'E' based the remaining dictionaries for the two words, but it could be skewed by the very high probability that the second word is ”THEORY.” Because a word of the form ”TH_O_Y” is very likely to be ”THEORY,” more information would be gained by focusing guesses based on the dictionary for the first word. To account for this, the value of information entropy was introduced into the weighting for each letter.

Each dictionary can be treated as a random variable representing the outcome for its respective word. When a dictionary has very low variability, such as the remaining dictionary that fit ”TH_O_Y,” the entropy will be very low, so little information can be gained. By multiplying the weight of each letter by the entropy of the dictionary for the word, the program can account for these scenarios. If one of the words is almost certain, then its entropy will be near zero, so the program will give it very little weight. By contrast, if very little is known about the word and its dictionary, its entropy will be high, so the program will place more weight on the letters from that

In summary, the weight on each letter was determined by the number of times it appeared in each dictionary, the frequency of the words it appeared in, and the entropy of each dictionary. To use a more mathematical notation,

$$\text{weight} = \sum_{\text{dict}} \sum_{\text{words}} (\#\text{ of times letter appears in word})(P(\text{word}|\text{dict})).(H(\text{dict}))$$

Each letter was weighted with this formula, and the one with the greatest weight was guessed. The dictionaries were refreshed based on the response to each guess, and the process was repeated until the phrase was found.