Hangman Strategy

Gobind Singh

July 4, 2023

1 Description

I tried a bunch of ideas, and the intuition that gave me the best success was to realize that probability distributions of letters appearing at a certain index holds fairly well across various dictionaries. I divided the training set into two parts and checked this.

Let P_x^i denote the probability of letter x appearing at position i in a word. Using a dictionary D, one can approximate this probability by counting the number of times the letter x appears at position i in all the words, and dividing that by the total number of words. The key idea here is that these probabilities transfer well to different dictionaries, which means you can calculate them on one dictionary and use them on another. As our problem is to solve for a disjoint dictionary, this was crucial.

Now on each guess, I try to maximise the expected number of hits in the word. For a letter x, the expected number of hits on guessing that letter will be

$$\sum_{i=1}^{length} P_x^i$$

where length is the total length of the word.

Apart from this once I have had a few guesses, I use the idea of adjusting my dictionary to approximate the P_x^i conditional on the correct and incorrect guesses I have had. However I use a context length idea here i.e the P_x^i is only influenced by all positions in $(i-c, i+c)$ where c is the context. What this means is to adjust my probability, I will use a conditional dictionary which will filter for any matches in the window $(i-c, i+c)$.

For example, to illustrate the process, let the word to guess is a 7 letter word and my context window is 2. And say I get two matches, "a" and "c" and now the word is " * * * a * c *". Now I will go and adjust P_x^i in the following way -

- For P_x^1, I see that I have no matches in the first 3 spaces (remember we look till the context window which is 2 and hence the final position to look for is 3). Hence my probabilities are unchanged.

- For P_x^2, I see that I have a match on the 4th space (remember we look till the context window which is 2 and hence the final position to look for is 4). So here I will filter my dictionary to get words which have "a" on the 4th position and then recompute P_x^2 using that dictionary.

Hence the core idea is at every step to recompute P_x^i is to look at c positions to the left and c positions to the right, and filter out from the dictionary words which match all the filled positions in those positions (i-c,i+c), and then recompute P_x^i.

This entire thing is done to not over fit to the training dictionary, and yet have some conditional information in my probability estimates. I tested and found that context window = 2 provides the best performance.

I found that my strategy gives a success rate of 48 % - 52 % on word subsets and practice games.