Documentation

Algorithrm

N-gram model is used which is based on conditional probabilities. The primary idea is that the model computes the probability of occurence of different letter tupes “l1 l2 l3…. ln” where n denotes the N chosen in n-gram model.

More advanced Neural Network models (LSTM) can also be deployed to solve the problem but it is easy to keep track of model working in this model and it is giving satisfactory results (>50% correct guesses)

Reasoning

Some of the parameters like the weight given to each n-gram, how many n-grams to work on and when to recalibrate the model

a. Picking 9 grams:

The average length of words in given dictionary is 9.3 and the 9 letter words were most occuring in the dictionary. Thus, the 9-gram model is chosen.

b. Weights to each of gram:

Chosing optimal weights is an optimisation problem but a common heuristic is to use the weights in propertion to the frequency of occurance. Hence, the weights were based on the probability of occurence of each n-letter word where words with length >=9 were clubbed within 9 letter word for probability calculations

c. Model recalibration

It is important to not include the words from the dictionary in probability calculation which are giving incorrect responses and are structurally different from given word. Hence, I created recalibration function to reset the dictionary for each gram.

I iterated on when to recalibrate the dictionaries for recalibrating when 1 mistake happens to never recalibrating. The primary issue with repeated calibration is that it takes significant time to run and the optimal results occured when recalibrated for <=3 mistakes.

Hence, in submission, the model is recalibrated when <=3 lives are left