**Trexquant ::** Hangman: Word Guessing Problem

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Strategy**

1. ***Dataset creation for Training***

*Most of the words have atleast ‘e’,’a’,’I’ in them so guessing with them would be efficient. And also a word with no letter known will be of less information for our trained model to guess so before passing a word to the model I have put a condition where atleast a single letter will be* known which will be decided by the length of word and the most frequent letter of that length in the training data.

Input shape – 38\*28

32 – for word till max length 32 and 6 for incorrect guesses. Each letter will be a vector of

28 , 26 for onehot encoding letter and 2 for flags of word length and incorrect guesses respectively.

Input to the model for training will be a random word selected from dictionary given and choosing the first guessed letter according to the probability. Next with some random probability we will display x% and y% of the correct guesses and incorrect guesses . Letters randomly selected for correct and incorrect guesses are according to the weights which are higher for more frequently occurring letters in dictionary.

**Change Done** : Letters are with probability 0.5 times selected randomly without taking the weights mentioned above and with 0.5 times taking the weights.And the model is trained for more number of epochs.

1. ***Model***

Tried training two sequence classification models, one Transformer Encoder and other LSTM where I found LSTM to perform well both in training as well while testing on the API runs. Here the sequence model output is passed through a FC layer with 26\*1 output shape for each letter where the target is 26\*1 vector with positions of ungessed correct letters to be 1.

Accuracy achieved in Recorded Games = **514/100**