Wordle Project A

Shrunjala Mul

Cost Function

$$Cost\ Function(word) = [\sum_{l=0}^{l=len(word)} l * P(l)] + [P(starting\ letter) + P(ending\ letter)]$$

Explanation : To get the most frequent occurring letters in the top guess was my idea behind using this cost function. The first half of the cost function is just the sum of all probabilities of letters in that word. This is the objective function. I wanted to maximize the probability of each letter so I get the most optimum guesses. The second half of this function is the regularization that I used to break the ties with. That is the probability of this starting letter and this ending letter of the word. This added an extra layer to break the ties.

Implementation : For the implementation I started by loading the data and counting the occurrences of each letter to then get the probability of each letter. Then I removed all the words with double lettered occurrences leaving us words with only unique letters. Then I proceeded to calculate the cost of each word with the function stated above considering the probability of the letters as well as the probability of the starting and ending letters. Then I picked the first choice as the one with the highest cost. I then moved on to removing that word from the list of next possible words and proceeded to calculate the next best choice. For this I first removed all the words containing the letters of the first choice and selected the one with the maximum cost after winnowing out a few words. By the third try I was left with only a few words to choose from and I repeated the process again to find the third best word.

First Word

The first word I found was **stare.** There were ties, which I broke by including the probability of the starting and ending letters too.

The top 20 choices for first best word were:
['stare', 'snare', 'slate', 'stale', 'arose', 'store', 'scare', 'arise', 'raise', 'irate', 'share', 'alert', 'crate', 'spare', 'saner', 'snore', 'alter', 'stole', 'trace', 'saute']

Second Word

After stare I had to pick a word that did not contain s, t, a, r, e. So I proceeded to remove them from the list and then used the cost function again to get the next word as **lingo.**

The top 20 choices for second best word were:
['lingo', 'login', 'logic', 'cloud', 'could', 'nobly', 'cling', 'limbo', 'lucid', 'blond', 'clown', 'dingo', 'doing', 'lying', 'clink', 'child', 'blind', 'ghoul', 'lunch', 'moldy']

Third Word

I followed the same process by removing all the words that contain s, t, a, r, e, l, i, n, g, o and was left with a small list of less than 20 words. The next best word was **duchy.**
The top choices for the third word were:
['duchy', 'chump', 'dumpy', 'mucky', 'jumpy']

In conclusion, My cost function follows all the criteria chalked out like it has 15 unique letters, it considers the probability of each letter occurring and also the starting and ending probabilities as the cost function and I get 3 good words to test out while playing wordle.