1 build_ngrams

The function receives a current dictionary and updates the dictionary ngrams. To be more accurate, ngrams is a list of dictionaries.

1. It receives a list of words.
2. Iterates through each word, and it considers sub-sequences of length between 1 to 4.
3. For each of these sub-sequences, it loops through letters of the sub-sequence and replace them with '.' and updates the values of ngram dictionary.
4. Keys of the ngram dictionary are tuples with format (letter, pattern) where pattern is a string.
5. For example [('a','.n.n.')] counts all the occurrences of ('anana') in all sub-sequences appearing in the words of the given dictionary.

2 get_probabilities

This function receives the pattern given in the clean_word from the guess function.

1. It loops through sub-sequences of clean_word with letters that are not discovered yet. It counts the number of patterns in ngram identical to this pattern.
2. It uses this frequency to estimate the probability of missing letters.

3 update_dictionary

This function receives the set of guessed_letters and the pattern of clean_word from the guess function. Its an attempt on conditioning the probabilities of the missing letters assuming the pattern of clean_word is already discovered.

1. It loops through the words in full_dictionary and takes on those words that are similar in structure to the pattern of clean_word. It penalize each word that deviate from this pattern, then takes on a fraction of word with least penalty.

2. In the case that the first few letters of the clean_word is missing (first four letters) it fills up the dictionary with four letter patterns that matches this pattern.

3. Then it calls build_ngrams to update the entries of the dictionary.

4 guess

It receives the word from start_game function.

1. First few steps it forces guesses that ensures at least one letter is found in every four consecutive letters

2. This amounts to guessing vowels first. (it can be justified from the training_dictionary, I added a code snippet at the end of my submission about this)

3. Until it reaches 3 tries left, it assumes that this word is part of the current_dictionary and it search for it inside the current_dictionary. At each step guessing letter that appears in most of the words in the current_dictionary that matches the pattern of clean_word.

4. It removes the entries from the dictionary that do not matches the pattern in clean_word (by considering letters that are correctly and incorrectly guessed.

5. when three tries left it works with ngrams probabilities.

6. When it reaches to two tries left it updates the dictionary.

7. At each iteration where one tries left, it updates the dictionary by calling update_dictionary.