Description of the strategy I deployed :-

- My strategy is statistical in the sense that first of all, I take maximum count of a letter in a word as one because ultimately, if a letter exists more than once it is filled at each location. I cleaned the word, removed spaces from input and searched for similar lettered word in dictionary and guessed the maximum occurring letter which is not guessed at current iteration.
- Next, I took out the ratio of vowels count to length of the world and found out that if 55% of the length of word is vowel, it is unlikely (from the distribution) that there will be more vowels.
- If we run out of similar words in given dictionary as the semi guessed word. So, like the thought process of a human, I try to break the word in smaller fragments (suffix/prefix) to try to fit in the most appropriate letter. This is what my algorithm tries to do by looking into another dictionary of the substring of words given in the original dictionary.
- At last if the letter is not found in the substrings' dictionary, the algo guesses the most common letter in whole words which is not yet guessed.