N-gram solver for Hangman

N-grams are continuous sequences of words or symbols, or tokens in a document. In technical terms, they can be defined as the neighboring sequences of items in a document. They come into play when we deal with text data in NLP (Natural Language Processing) tasks.

For example, given the word “Hello”, we have:
Unigram: “H”, “e”, “l”, “l”, “o”
Bigram: “He”, “el”, “ll”, “lo”
Trigram: “Hel”, “ell”, “llo”
Fourgram: “Hell”, “ello”
Fivegram: “Hello”

Given a "broken" word, which means with “_”, the algorithm goes to its training set to find a similar case and count the frequency of each letter. Based on this frequency, the prediction is determined.
There are also different cases of "similarity". For example, when we think of a two-letter combination (bigram), we calculate the frequency of, say, "_a," in which case the space is 'a', the frequency of 'b', and so on.
When considering three-letter combinations (trigram), we calculate such frequencies as the probability of "a" in the "f_r" case, the probability of "b", and so on. And so on, we'll consider up to five letter combinations.
Also, I regard fivegram is much more preferable than fourgram since when we know 4 letters and one blank, this has more information than three letters and one blank. Thus, the weights fom unigram to fivegram are given from smallest to largest correspondingly.

Fivegram_scores(word)
Cases:
'_cccc' case
'c_ccc' case
cc_cc' case
'ccc_c' case
'cccc_' case
'c__cc' case
'cc__c' case
guess(fivegram_scores('a__le'))

fourgram_scores(word)
Cases:
'c__c' case
'ccc_' case
'cc_c' case
'c_cc' case

'_ccc' case
guess(fourgram_scores('f__r'))

Trigram_scores:
Cases:
'_cc' case
c_c' case
'cc_' case
'__c' case
'c__' case

Bigram_scores:
Cases:
'_c' case
'c_' case

Finally, the practice accuracy is about 60%.
The recording accuracy of 1000 times is **55.1%**.