Dessert Labs Coding Challenge: Word Cooccurrence

Evaluation:
We are looking for people who can write great code, solve problems correctly, and do so rapidly. The following is a genuine problem that you might need to work on, and we'll be assessing you on both correctness, and speed.

If your program gives the correct answer for our test input, then we'll look at how quickly you finished your solution. There is a 1.5 hour time limit, but really good programmers can do this in less.

Background:
Word cooccurrence is a way of determining how commonly two words appear together in a language. For example, the words "for" and "example" appear more commonly together than "dessert" and "labs" in common english.

One way of defining cooccurrence is: for each occurrence of word A, how often does word B appear within K words?

For example, in the sentence "*I like dessert, and I like Dessert Labs.*", if A=dessert, B=like, and K=3, then both occurrences of "*dessert*" have the word "*like*" within 3 words range. So the **cooccurrence**(A="dessert", B="like", K=3) = 2. (Note that the first "*dessert*" had the word "*like*" within ±3 words' range twice, but we only count this once.)

The **cooccurrence probability** is then defined as:

$$p(A,B,k) = \frac{cooccurrence(A,B,k)}{count(A)}$$

This can be thought of as: "out of every time the word A appears, how many of those times does B appear within k words".

For example:

$$p(\text{"}dessert\text{"}, \text{"}like\text{"}, 3) = \frac{2}{2} = 1$$
$$p(\text{"}dessert\text{"}, \text{"}labs\text{"}, 3) = \frac{1}{2} = 0.5$$
$$p(\text{"}labs\text{"}, \text{"}dessert\text{"}, 3) = \frac{1}{1} = 1$$

Task:
Given an input document, write a program to calculate all cooccurrence probabilities for a document and given range K. Your program will then be given multiple input lines from stdin with one word pair A and B per line; for each input line, your program should print the cooccurrence probability of A, B.

- Your program must **load and process the input document on startup**; you should do any expensive processing at this point.
- Your program will **then receive word pairs on standard input**. Output must be fast!
- K will be an integer with 1 <= K <= 5.
- Output should be printed **rounded to 2 decimal places**.

- Case of words should be ignored, and all whitespace and punctuation stripped.
- You may assume all characters are ASCII, and that the filename provided is valid.
- If word A does not appear in the text, you should output 0.00.

Performance Requirements:

- Initial processing of the input document should be **O(NK)** worst case time complexity, where N is the number of words in the input document.
- Results for each pair of words needs to be produced in **O(1)** worst case time complexity.
- **Your solution needs to be both correct and meet the above performance requirements in order to pass!**

Example 1:

e.g. if your program is in Python, it would be run with the following command and standard input:

```
$ python cooccurrence.py dessert.txt 3
dessert like
dessert labs
labs dessert
```

With dessert.txt:

```
I like dessert, and I like Dessert Labs.
```

Expected standard output for the above input:

```
1.00
0.50
1.00
```

Example 2:

Download the data file http://dssrt.com/cat-in-the-hat.txt.

```
$ python cooccurrence.py cat-in-the-hat.txt 3
hat cat
cat hat
sit sit
dessert labs
```

Expected standard output for the above input:

```
0.71
0.38
0.80
0.00
```