CS 490 FP
Programming assignment 2

Who is the author of this document?

A common problem in historical and literary scholarship is determining who the author of a particular document was. This can also be important in detecting plagiarism or ghost-writing.[1]

We begin with a *corpus* of known text. We use this to build up a 'profile' of how this author uses language—specifically, the relative frequency with which they use various words. Then, given a text of uncertain authorship, we compare its profile to those of known authors—the one with the highest similarity (or, equivalently, smallest difference) is taken as the most likely author.

You are given two known texts of about the same size—H. P. Lovecraft's novella *The Case of Charles Dexter Ward* and Arthur Conan Doyle's *The Hound of the Baskervilles.* (These were chosen mostly because they're public domain and were readily available online. There's nothing special about them as texts.) You also have two other files that we're pretending are of uncertain authorship ("Cool Air" and "The New Catacomb"); your program should predict which author wrote which short story.

You will need to write a program that does several things. Build it a bit at a time, in this order. Remember to think functionally; use the output of one function as input to the next; remember the design patterns. I'd suggest focusing on one task at a time, and then, as each is solved, putting it into a function so you can do the same thing with several files. (You'll need to develop profiles for 4 files, and compare each "mystery" story with 2 different profiles).

Put the text files into the same folder as your program file.

Profiling a text:

1. Open the file and read the text as a single string, closing the file when finished.
2. Clean the string you just read: The string should be converted to all one case—upper or lower doesn't matter, as long as you're consistent throughout the program. All instances of "\n", quotation marks, commas, periods, question marks, exclamation marks, hyphens, dashes, colons and semicolons should be replaced by spaces. Note that single quotes should be left alone, as the apostrophe indicates the possessive (Bill's book, Mary's hat, etc). This can be done via multiple calls to a string-replace function or by use of regular expressions. (Repeatedly calling string functions will be faster to write; regular expressions will result in more compact code.)
3. Split the string on whitespace to produce a list of strings (individual words).
4. Use this list to produce a hash with strings (words) as keys and counts as values; that is, for each word, the value is the number of times that word appears in the text.
5. Find the sum of all values, and divide each word count by the total, so the values are now the relative frequency of each word, that is, the probability that a randomly-chosen word is this one.
6. You'll notice that these are fractions or floating-point numbers, almost all 1/1000 or less. This is going to be awkward. We're going to convert this relative frequency into a value that's easier to

1 One of the *Federalist Papers*, one of the basic documents of American democracy, was of uncertain authorship until fairly recently. It was known to be one of two authors...but which? The method we're using here was key in identifying the actual author. (One of them was more likely to use the British *whilst* than the Colonial (i.e. American) *while*.) Likewise, it was known that at least the last few of L. Frank Baum's *Wonderful Wizard of Oz* books were ghost-written, but it wasn't known where the ghost-writing began; as it turns out, several of the books that had been thought to be purely Baum's work were partially ghost-written.

work with. Specifically, for each frequency (value), take the base-10 logarithm[2], and multiply by -1. This will convert your frequency measures into positive numbers, mostly between about 3.1 and 4.8, with higher values corresponding to increasing rarity. That is, the higher this value, the less likely this author is to use this word. *This is what your main program will need. If all of your other list and hash manipulation was in functions, then when you return this to your main program, all of your earlier versions become eligible for garbage collection.*

7. To compare two profiles: For each word in one profile, if that word is present in the other profile, find the absolute value of the difference in their frequency scores. The overall similarity is the average of these differences. Note that by this measure, more similar texts have *lower* scores; higher scores indicate more *difference* between them.
8. To predict the most likely author: Compare the profile of each 'mystery' text with the profile of each other. The author with the lowest profile comparison (as defined above) is predicted to be the most likely author.

For this program, you can hard-code file names in where necessary. Part of your grade will be on program efficiency and use of a functional style.

For comparison, here's the output of my program:

```
Welcome to DrRacket, version 8.5 [cs].
Language: racket, with debugging; memory limit: 128 MB.
Analyzing Mystery Text 1...
        Mystery Text 1 is probably Lovecraft
Analyzing Mystery Text 2...
        Mystery Text 2 is probably Doyle
>
```

And, in this case, the program is right on both.[3] Of course, these are the only authors the program knows about. We could give it a passage from Shakespeare or an article from today's news, and it would classify the text as being from one or the other.

For comparison, my implementation, which could be tightened up in a couple of places, runs to about 80 lines of source code. There's not a lines-of-code limit, but if your program is running into pages and pages of code, you may want to step back and rethink your approach.

Submit your source code or GitHub link to Canvas by the deadline.

---

2 This is mostly for ease of interpretation. If two authors use the same word, but their frequency measures differ by about 1.0, then one author uses that word about 10 times as much as the other author. (And it can be that much; for Lovecraft and Doyle, a few words differ as much as about 1.7, meaning one author uses that word $10^{1.7}$, i.e. about 50 times as much as the other.) Our method would work just as well with natural logarithms, but wouldn't have as intuitive an interpretation.

3 As to the question originally posed—Who is the author of this document? *This* document was written by Prof. Hare, obviously.