

Due: 08/17/2007 @ 5:00pm

Name:						
Student ID:				Instructor:	Borja Sotomayor	
				Do not write i	n this area	
	1	2	3	4	TOTAL	
				Maximum possil	ple points: 45	



Due: 08/17/2007 @ 5:00pm

Exercise 1 <<15 points>>

Implement the hangman game (Exercise 4 from Homework #2, including the extra credit portion) using Python.

Exercise 2 <<20 points>>

In language processing, two texts can be compared to determine how similar they are according to multiple criteria. In this exercise, we will simplify the definition of "distance" between two texts, in such a way that it will be possible to compute this distance using a python script.

Before computing the distance, we need to generate the following information for each text:

- Word frequencies. Given the text, compute the frequency of each word. This should result in a file containing a list of (word, frequency) pairs, which we will denote as (W_i, f_i) . You are already provided with several example frequency files for this exercise. Writing a word frequency generator is a separate exercise.
- Normalized word frequencies. Given the word frequencies, normalize them. To do this, simply take each frequency and divide it by the square root of the sum of the squares of all the frequencies:

$$(w_i, \frac{f_i}{\sqrt{\sum f_i^2}})$$

You can consider the normalized word frequencies as the specification of a point, where each word is a dimension, and the normalized frequency is the position of the text on that dimension. So, we compute the distance between two texts as a Euclidean distance:

$$distance(text_1, text_2) = \sqrt{\sum (fnorm_{1,i} - fnorm_{2,i})^2}$$

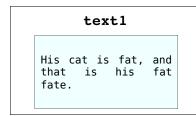
Note: If a text has a word that never appears in the other text, you can consider that the position along that word-dimension is 0 in the other text.

You must write a Python script called **distance.py** that, given two files (specified as parameters) with the word frequencies of two texts, computes the distance between the two texts.

Due: 08/17/2007 @ 5:00pm

Example

Suppose we have the following two files:



t	ext2	
is	is fat, and his feline	

The word frequency files would be:

text	l.freq	
fat his is and cat fate that	2 2 2 1 1 1 1	

text2
is and cat fat fate feline his that

To compute the distance between the two files, we would run our program like this:

python distance.py text1.freq text2.freq



Due: 08/17/2007 @ 5:00pm

Internally, the **distance.py** program would first compute the normalized frequencies, which would be the following:

<pre>text1.freq (Normalized)</pre>		
fate cat and fat that is his	0,25000 0,25000 0,25000 0,50000 0,25000 0,50000	

(Norm	nalized)
feline fate the cat and fat that his	0,28868 0,28868 0,28868 0,28868 0,28868 0,28868 0,28868 0,28868 0,57735

Based on these frequencies, and using the distance formula, the script would output the following:

0.517641

You are provided with several word frequency files you can use to test your solution:

- Distance between shrew.txt.freq and muchado.txt.freq: 0.358753109402
- Distance between shrew.txt.freq and baskervilles.txt.freq: 0.60759609533
- Distance between shrew.txt.freq and oz.txt.freq: 0.696044403252
- Distance between shrew.txt.freq and rfc2821.txt.freq: 0.884945364129

Exercise 4 <<10 points>>

Write a word frequency generator in Python. Your program must take a text file and count the number of times each word appears. Of course, no text is just a sequence of words, and there is also punctuation, spacing, etc. to take into account. So, for simplicity, our word frequency generator will be *case-insensitive* and will consider *any non-letter character* as a separator between words. So, for example, "the dog's nose" would be considered as four words: "the", "dog", "s" and "nose". Similarly, "The year 1991 was an uneventful one" is considered as six words: "The", "year", "was", "an", "uneventful", "one" (notice how the number 1991 is ignored, as it is just considered part of the separation between "year" and "was").



Due: 08/17/2007 @ 5:00pm

Your program must be written in a file called **freq.py**. The program will accept two command line parameters: the file whose word frequencies we will determine, and a file in which to store the word frequencies.

python freq.py example.txt example.txt.freq

The output file will have a line for each word, consisting of the word, a space character, and the frequency. You should be able to use the generated file with your solution to Exercise 3. Note, however, that your generated file need not be *exactly* the same as the provided ones (these are sorted by frequency, and then by word; your file does not need to be sorted).

Hint: The solution to this exercise can be greatly simplified if you use the regular expressions module of the Python Standard Library.