Assignment 6 PartII

CS 2000: Python Programming Language

Wassnaa AL-mawee

Western Michigan University

April 13, 2018

In this assignment, you will create a spell-checking program in Python 3. It is worth 5 points and is due (04/20/2018), 12:00 pm.

1. Description

For this assignment we will be creating a program which computes the Levenshtein distance between two strings using the decorator based memoization technique. The program should be a stand-alone Python program (i.e. callable from the command line). There will be three main sections of code:

1) `lev(a, b)` **naive Levenshtein distance recursive function**: This is the inefficient function presented initially in class. We will be using decorators to improve the performance of this function so you should not need to modify the original. If you need a reference the Wikipedia article has a nearly verbatim pseudocode at https://en.wikipedia.org/wiki/Levenshtein_distance

2) **MemoizeReset(f) automatic resetting memoization decorator**: We will create a decorator which memoizes our Levenshtein distance recursive function. We saw several examples of this in class, however for the purposes of this assignment we have several needs which will prevent us from using the exact same technique presented in class. First, the function will need to reset its cache after each external function call automatically (i.e. you should not require the external caller to call `lev.reset()` or some such after each call). This resetting behavior is necessary to prevent the cache from growing to ridiculously large proportions while at the same time allowing us to avoid recomputing large portions of the recursion tree. Secondly, to demonstrate that this is working properly add functionality to your decorator so that it also returns the number of times the function is called for each distance computation.

3) **main**: Your program should be callable from the command line. It should function in the following way:

   - `python lev.py -f filename`: Read a file (given by filename) consisting of lines of the form:

     `word1, word2`

     and for each line print the Levenshtein distance between the two words and the number of function calls required for the Levenshtein distance computation.

   Your program should fail gracefully (i.e. print an informative error message and exit()) under the following conditions:

- The number of arguments is incorrect.
- The file can’t be found or is formatted incorrectly.

2. Data file and output

- Download the data file wordfile.txt. It contains 30 lines, each with two words.
- Please format your output in the following manner:

# Output:
word1a, word1b, lev(word1a, word1b), numcalls1
.
.
.
.
word30a, word30b, lev(word30a, word30b), numcalls30

3. What to turn in:

(Remember READABILITY COUNTS!)

- On e-Learning, submit your Python file whose name <hw#6PartII_LastName.py>. Please format your Python code in the following manner:

# Name: <your name here>

# Date: <#/#/#>

# Homework: <#>

# Your code