CSE 231 Fall 2009

Programming Project #7

Assignment Overview

This project focuses on Dictionary and Input/output file manipulation, as well as more experience on writing functions. It is due Monday, November 2nd before midnight. The application is to design a spell-checking tool that interactively corrects misspelled words in a document. The user to selects the correct spelling of a misspelled word from a list of candidates provided by the program. This project is worth 50 points.

The Problem

The **spell checking tool** will perform three tasks:

1- Spot the misspelled words in a file by checking each word in the file against a provided dictionary.
2- Provide the user with a list of alternative words to replace any misspelled word.
3- Write a new file with the corrected words as selected by the user. ***Note***: It would take some work to maintain the original file's punctuation. Don't worry about that for this project.

A list of correctly spelled English words is provided as a reference, one word per line of the file. The user file to be checked is text file where each space-separated word of the file is to be examined. As previously, punctuation and case should be ignored. If a word from the file does not exist in the provided list then it is assumed to be misspelled and a set of alternatives provided to the user. The user has three options for the provided candidate list:

- select, by number, one of the candidates. The word in the file is replaced in the output file by the selection
- indicate that they wish to leave the word "as is"
- indicate that they wish to provide the alternative by typing it in directly. The program replaces the word in the file with the user provided replacement in the output file.

You need to set some rules to select the best word candidates. We provide the following rules but we are open to new rules that might increase the accuracy and efficiency of this tool (see below).

- Consider candidates from the reference list that are +/- *n* in size with the misspelled word. n is a parameter you may experiment with to find a good result
- Check that the first *n* letters of the two words are the same where n is a parameter you can experiment with to find a good value.
- Check that the number of common letters between the word and a candidate are within some parameter *n* (also to be checked).
- Check the number of letters that are in the same location for the word and a candidate. Again find an appropriate parameter for the limit.

Extra Credit (10 pts)

You may think of other, better strategies for spell checking. If you do, please indicate in your comments that you are using augmented rules of your own design. Obviously, you should explain the rules in comments and they should perform better than the above rules. You could also try to maintain punctuation. Extra credit is all or nothing. You either get all the points or none.

Program Specifications

You project will do the following:

1. Prompt the user for the name of the file to spell check. The program will spell check each word and then write a new file with the name of the original file plus the text "-chk". Thus if the file being checked is "file.txt", the spell checked output will be "file-chk.txt". Note that the file suffix is preserved!
2. Prompt the user for the name of the reference list. It should default to "ref.txt". This file consists of words, one word per line.
3. Create a function that will read the reference list words from a file and save them in an appropriate data structure.
4. Create a function that checks a word of the user file against a word from the reference list. The rules mentioned above are to be implemented in this function. This function either returns False, meaning that the word was in the reference list, a list of candidate replacements or an empty list if it could not find a candidate.
5. If the word does not exist in the reference list, print a list of all candidate words for the user with an index for each candidate. If the user enters the index of a candidate word, the original word is replaced in the output file with the candidate. If the user enters a -1, then the word is not replaced. If the user enters -2, then the user will type in a replacement and that replacement will be used in the new output file.
6. You must provide error checking for the user if they enter an improper index (bigger than the list of candidates, smaller than -2).
7. Output the updated spell-checked file.

Notes:

- the lower() function is important when you read each word, because no match case is required in this assignment.
- The str() , list() functions are also useful here.
- The split() function is important to slide a line into a list.
- The strip function is good to get rid of unnecessary characters attached to a word, such as " , /)(." and all other punctuations.
- The use of sets (for finding common words) and dictionaries (for lookup) would be useful here.
- This will take some experimenting. Implement the main program then test each function, especially the one that returns the candidate list.
- Again, maintaining punctuation is extra work and can be ignored.

Deliverables

proj07.py -- your source code solution (remember to include your section, the date, project number and comments).

1. Please be sure to use the specified file name, i.e. "proj07.py"
2. Save a copy of your file in your CS account disk space (H drive on CS computers).
3. Electronically submit a copy of the file.

Example

Below is an example interaction with the provided test file sample.txt. You do not have to meet the standards of the below example but it shows you what you can do. Experiment, see what you can do.

```
main()
Commands:
-1 take the word as is
-2 prompt for the  replacement
0 to len(w)-1: the selected replacement

What refernence list (ref.txt default):
What file to check?:sample.txt
scientsits : 0:scientific,1:scientist,2:scientists,
Action:2
ouput : 0:ought,1:output,
Action:1
intensiyt :
0:incentives,1:incessant,2:ingenuity,3:insensibly,4:integrity,5:inte
llects,6:intensely,7:intensify,8:intensity,9:intensive,10:intentions
,11:intently,12:intentness,13:intercepts,14:intercity,15:interjects,
16:interments,17:interprets,18:interrupts,19:intersect,20:intersects
,21:interviews,22:intestinal,23:intestine,24:intestines,25:intrusive
,26:inventions,
Action:8
40 :
Action:-1
avg : 0:ave,1:aver,2:avid,3:avow,
Action:-2
Replacement:average
temprature : 0:temperature,
Action:0
All done
```