

Assignment 2

Spell Checker

Introduction

Please design a spelling checker to examine whether an input word is correctly spelled or not. If the word is misspelled, please list its nearest words which have the computed editing distances smaller than or equal to 3 from the given dictionary. (Remember to list the words in the ascending order of editing distance) Your dictionary file is here ([dictionary.txt](#)). The materials you need to upload is the same as those in the last programming assignment.

Process

First, the dictionary file was read in and the words in the dictionary were collected. Next the word is accepted and the dictionary is searched to see if it is in it. A set data structure of the words was created for faster searching. If the word is found in this set of words then a message saying it is correctly spelt is printed.

If the word is incorrectly spelt, the Edit Distance (or Levenstein Distance) is calculated for each word in the dictionary. The Edit distance was calculated based on the algorithm below

- The cost $D(0, 0)$ of the node $(0, 0)$ is zero.
- Each node (i, j) can be reached only through three allowable predecessors $\{(i-1, j), (i-1, j-1), (i, j-1)\}$.

- **Diagonal transitions:**

$$d(i, j|i-1, j-1) = \begin{cases} 0 & \text{if } \mathbf{r}(i) = \mathbf{t}(j) \\ 1 & \text{if } \mathbf{r}(i) \neq \mathbf{t}(j) \end{cases}$$

- **Horizontal and vertical transitions:**

$$d(i, j|i-1, j) = d(i, j|i, j-1) = 1$$

- Algorithm

```

D(0,0)=0
for i = 1 to I
    D(i,0)=D(i-1,0)+1
for j = 1 to J
    D(0,j)=D(0,j-1)+1
for i = 1 to I
    for j = 1 to J
        c1 = D(i-1,j-1)+d(i,j|i-1,j-1)
        c2 = D(i-1,j)+1
        c3 = D(i,j-1)+1
        D(i,j) = min(c1, c2, c3)
D(A,B) = D(I,J)

```

To help speed up this process, the editing distances of words that are too long or too short are passed over. This is because if a word is too long or too short, its editing distance can't possibly be less than the set editing distance limit. For example, given an input string "wosked" and an editing distance limit of 3, then it is clear that the string "woskedaaaa" and "wo" both would have an editing distance of 4 or higher due to insertions or deletions, so it does not make any sense finding the editing distances for these strings. They will not be included as suggestions anyway.

After the editing distances are calculated for words they are placed in a list based on their editing distance value. Words with editing distances of 1 are placed in the first list, words with editing distances of 2 are placed in the 2nd list, etc. This was done for easier printing after.

Finally the suggested words are printed. The words in the 1st list are printed first then those in the second list are printed and so on until all suggestions are printed.

Results

Non-Case Sensitive Spell Checker results

```
C:\windows\system32\cmd.exe
C:\MachineLearning\Spell Checker>python spellChkerNONCASESENSITIVE.py
NON-CASE SENSITIVE Spell Checker (type 'end' to end)
Word: wellcome

1. welcome

2. become                3. bellicose

NON-CASE SENSITIVE Spell Checker (type 'end' to end)
Word: potatoes

1. potatoes

2. Coates                3. notate                4. penates
5. potato                6. Potts                7. rotate

8. Antares              9. apostate              10. Bootes
11. borate              12. donate              13. estate
14. Gates              15. gotten              16. iodate
17. loaves              18. locate              19. Lotte
20. mutate              21. mutatis              22. nutate
23. oases              24. orate                25. otter
26. ovate              27. palate              28. pate
29. pater              30. petite              31. Pilate
32. pirate              33. plate                34. platen
35. Polaris            36. polite              37. pomade
38. portage            39. Porte                40. postage
41. potable            42. potash              43. potlatch
44. pottery            45. probate             46. prolate
47. prorated           48. prostate            49. pupate
50. rotten             51. Socrates            52. state
53. Staten             54. stater              55. status
56. tate               57. tater               58. tomatoes
59. upstate            60. upstater            61. vocate
62. Yates
```

```
NON-CASE SENSITIVE Spell Checker (type 'end' to end)
Word: WELcome
Word is correctly spelt
```

```
NON-CASE SENSITIVE Spell Checker (type 'end' to end)
Word: WELLCOME
```

```
1. welcome

2. become                3. bellicose
```

```
NON-CASE SENSITIVE Spell Checker (type 'end' to end)
Word: end
```

Case Sensitive Spell Checker results

```
C:\windows\system32\cmd.exe
C:\MachineLearning\Spell Checker>python spellChkerCASESENSITIVE.py
CASE SENSITIVE Spell Checker (type 'end' to end)
Word: wellcome

1. welcome

2. become                3. bellicose

CASE SENSITIVE Spell Checker (type 'end' to end)
Word: potatoes

1. potatoes

2. Coates                3. notate                4. penates
5. potato                6. rotate

7. Antares              8. apostate              9. Bootes
10. borate              11. donate               12. estate
13. Gates               14. gotten               15. iodate
16. loaves              17. locate               18. Lotte
19. mutate              20. mutatis              21. nutate
22. oases               23. orate                24. otter
25. ovate               26. palate               27. pate
28. pater               29. petite               30. pirate
31. plate               32. platen               33. polite
34. pomade              35. portage              36. postage
37. potable             38. potash               39. potlatch
40. pottery             41. Potts                42. probate
43. prolate             44. prorate              45. prostate
46. pupate              47. rotten               48. Socrates
49. state               50. Staten               51. stater
52. status              53. tate                 54. tater
55. tomatoes           56. upstate              57. upstater
58. vocate              59. Yates
```

```
CASE SENSITIVE Spell Checker (type 'end' to end)
Word: WELcome

1. become                2. come                  3. Ecole
4. income                5. Lome                  6. upcome
7. welcome

CASE SENSITIVE Spell Checker (type 'end' to end)
Word: WELCOME
No suggestions found.

CASE SENSITIVE Spell Checker (type 'end' to end)
Word: end
```

Discussion

The only issue with this assignment is the runtime of the program when finding suggestions. I believe 1 way to lessen the runtime would be to initially sort the words by length, instead of checking for length each time. That way, I won't have to run through the entire dictionary each time looking for words of a certain length first.

Summary

The assignment was fairly simple. The concept of Levenshtein Distance was also easily understood. This project was a fun little assignment to put that knowledge into practice.