https://github.com/norvig/pytudes/blob/master/py/spell.py

"""Spelling Corrector in Python 3; see http://norvig.com/spell-correct.html

Copyright (c) 2007-2016 Peter Norvig

MIT license: www.opensource.org/licenses/mit-license.php """

################ Spelling Corrector

import re

from collections import Counter

def words(text): return re.findall(r'\w+', text.lower())

WORDS = Counter(words(open('big.txt').read()))

def P(word, N=sum(WORDS.values())):

"Probability of `word`."

return WORDS[word] / N

def correction(word):

"Most probable spelling correction for word."

return max(candidates(word), key=P)

def candidates(word):

"Generate possible spelling corrections for word."

return (known([word]) or known(edits1(word)) or known(edits2(word)) or [word])

def known(words):

"The subset of `words` that appear in the dictionary of WORDS."

return set(w for w in words if w in WORDS)

def edits1(word):

"All edits that are one edit away from `word`."

letters = 'abcdefghijklmnopqrstuvwxyz'

splits = [(word[:i], word[i:]) for i in range(len(word) + 1)]

deletes = [L + R[1:] for L, R in splits if R]

transposes = [L + R[1] + R[0] + R[2:] for L, R in splits if len(R)>1]

replaces = [L + c + R[1:] for L, R in splits if R for c in letters]

inserts = [L + c + R for L, R in splits for c in letters]

return set(deletes + transposes + replaces + inserts)

def edits2(word):

"All edits that are two edits away from `word`."

return (e2 for e1 in edits1(word) for e2 in edits1(e1))

################ Test Code

def unit\_tests():

assert correction('speling') == 'spelling' # insert

assert correction('korrectud') == 'corrected' # replace 2

assert correction('bycycle') == 'bicycle' # replace

assert correction('inconvient') == 'inconvenient' # insert 2

assert correction('arrainged') == 'arranged' # delete

assert correction('peotry') =='poetry' # transpose

assert correction('peotryy') =='poetry' # transpose + delete

assert correction('word') == 'word' # known

assert correction('quintessential') == 'quintessential' # unknown

assert words('This is a TEST.') == ['this', 'is', 'a', 'test']

assert Counter(words('This is a test. 123; A TEST this is.')) == (

Counter({'123': 1, 'a': 2, 'is': 2, 'test': 2, 'this': 2}))

assert len(WORDS) == 32198

assert sum(WORDS.values()) == 1115585

assert WORDS.most\_common(10) == [

('the', 79809),

('of', 40024),

('and', 38312),

('to', 28765),

('in', 22023),

('a', 21124),

('that', 12512),

('he', 12401),

('was', 11410),

('it', 10681)]

assert WORDS['the'] == 79809

assert P('quintessential') == 0

assert 0.07 < P('the') < 0.08

return 'unit\_tests pass'

def spelltest(tests, verbose=False):

"Run correction(wrong) on all (right, wrong) pairs; report results."

import time

start = time.clock()

good, unknown = 0, 0

n = len(tests)

for right, wrong in tests:

w = correction(wrong)

good += (w == right)

if w != right:

unknown += (right not in WORDS)

if verbose:

print('correction({}) => {} ({}); expected {} ({})'

.format(wrong, w, WORDS[w], right, WORDS[right]))

dt = time.clock() - start

print('{:.0%} of {} correct ({:.0%} unknown) at {:.0f} words per second '

.format(good / n, n, unknown / n, n / dt))

def Testset(lines):

"Parse 'right: wrong1 wrong2' lines into [('right', 'wrong1'), ('right', 'wrong2')] pairs."

return [(right, wrong)

for (right, wrongs) in (line.split(':') for line in lines)

for wrong in wrongs.split()]

if \_\_name\_\_ == '\_\_main\_\_':

print(unit\_tests())

spelltest(Testset(open('spell-testset1.txt')))

spelltest(Testset(open('spell-testset2.txt')))