spellingcorrector

January 19, 2022

A simple spelling corrector Inspired by the one found on Peter Norvig's page (https://norvig.com/spell-correct.html). It finds the correction c maximizing the conditional probability P(c | w) that c is the "right" word given that the user entered w.

By Bayes law: P(c | w) = P(w | c) P(c) / P(w)

About these probabilities:
P(w | c) is modeled as a function of edits needed to transform w into c. The number of edits is supposed to follow a binomial distribution.
P(c) is the "language model", telling us how much c is likely to occur. This is estimated from data.
P(w) is the same for all c and can be ignored.

The file big.txt is used to estimate P(c). It can be downloaded from Norvig's page: https://norvig.com/big.txt
Any other large file of text could be used instead.

[3]:
```
# Necessary imports.
import sys
import math

# Read the source file content.
# Return a dictionary mapping (lowercase) words to probabilities
def read_words(filename):
    counters = {}
    total = 0
    f = open(filename)
    for line in f:
        for word in line.lower().split():
            # exclude strings including non-alphabetic characters
            if word.isalpha():
                if word in counters:
                    counters[word] += 1
                else:
                    counters[word] = 1
                total += 1
    probs = {}
    for word in counters:
        probs[word] = counters[word] / total
    return probs
```

[4]:
```
# Edits for the given word.
def edit1(word):
    # Set of single editings starting from the given word.
    n = len(word)
    letters = "abcdefghijklmnopqrstuvwxyz"
    variations = set()
    # Deletions
    for i in range(n):
        newword = word[:i] + word[(i + 1):]
        variations.add(newword)
    # Substitutions
    for i in range(n):
        for l in letters:
            if l != word[i]:
                newword = word[:i] + l + word[(i + 1):]
                variations.add(newword)
    # Insertions
    for i in range(n + 1):
        for l in letters:
            newword = word[:i] + l + word[i:]
            variations.add(newword)
    # Inversions
    for i in range(n - 1):
        newword = word[:i] + word[i + 1] + word[i] + word[(i + 2):]
        variations.add(newword)
    return variations
```

[5]:
```
# Set of editings obtained with k operations from the given word.
def edit_k(word, k):
    variations = {word}
    for _ in range(k):
        newvars = set()
        for v in variations:
            newvars |= edit1(v)
        variations = newvars
    return variations

# Binomial coefficient 'n choose k'.
def binomial_coefficient(n, k):
    denominator = math.factorial(k) * math.factorial(n - k)
    return math.factorial(n) // denominator

# Formula to count P(w|c) given n, e, q - probability of e errors on a word of␣
 ↪length n.
def probability_error(n, e, q):
    binc = binomial_coefficient(n, e)
    return binc * (q ** e) * ((1-q) ** (n - e))
```

[6]:
```
# Spelling corrector.
def correct_word(word, dictionary):
    q = 0.01
    candidates = []
    for e in range(3):
        variations = edit_k(word, e)
        for c in variations:
            if c in dictionary:
                p_c = dictionary.get(c)
                p_wc = probability_error(len(c), e, q)
                score = p_wc * p_c
                candidates.append((score, c))
    candidates = candidates[:5]
    candidates.sort(reverse=True)
    return candidates

dictionary = read_words("big.txt")
word = input("Enter a word: ").lower()
candidates = correct_word(word, dictionary)
for score, c in candidates:
    print(c)
print(candidates)
```

```
Enter a word: frnak
frank
freak
creak
prank
crank
[(1.2059752748332442e-06, 'frank'), (1.0486741520289079e-07, 'freak'),
(7.414867741618542e-09, 'creak'), (3.1778004606936607e-09, 'prank'),
(1.0592668202312202e-09, 'crank')]
```

[ ]: