How to Write a Spelling Corrector

Dongqing Zhu

Based on Peter Novig's Online Tutorial
http://norvig.com/spell-correct.html

Introduction

- Peter Novig's Online Tutorial
  - Exellent material for HW3
  - Simple probability methods
  - 21 lines of python code
  - ~ 6.5 MB training data
  - ~ 70% correctness on two test datasets

Outline

- Problem Formulation
- Solving the Problem
- Evaluation
- Improvement
- Useful Links

Problem Formulation

- Problem Statement
  - Find the most likely spelling correction **c** for a given word **w** among all possible corrections

- Probability Theory
  - argmax$_c$ P(c|w)

    => argmax$_c$ P(w|c) P(c) / P(w) -- by Bayes' Theorem

    => argmax$_c$ P(w|c) P(c) -- by ignoring P(w)

Solving the Problem

- argmax_c P(w|c) P(c)
  - argmax_c
    - cover all possible corrections ideally
  - P(c)
    - Language Model
    - capturing how likely **c** would stand on its own
  - P(w|c)
    - Error model
    - capturing how like the change **c** => **w** will happen

argmax_c

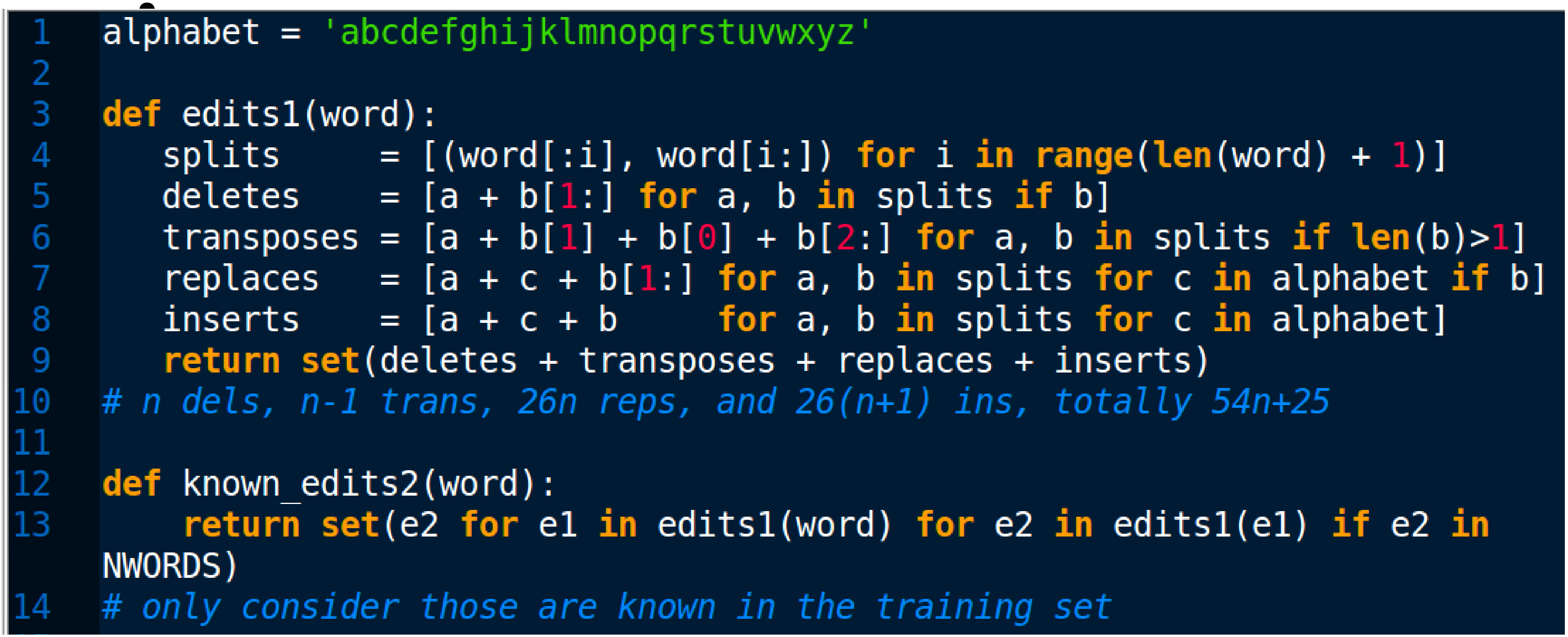
- Enumerate possible corrections by edit distance
  - words of edit distance 1

    cover 80% to 95% of spelling errors
  - words of edit distance 2

    cover 98.9% of spelling errors

argmax$_c$

- Get possible corrections by edit distance



```python
alphabet = 'abcdefghijklmnopqrstuvwxyz'

def edits1(word):
   splits     = [(word[:i], word[i:]) for i in range(len(word) + 1)]
   deletes    = [a + b[1:] for a, b in splits if b]
   transposes = [a + b[1] + b[0] + b[2:] for a, b in splits if len(b)>1]
   replaces   = [a + c + b[1:] for a, b in splits for c in alphabet if b]
   inserts    = [a + c + b     for a, b in splits for c in alphabet]
   return set(deletes + transposes + replaces + inserts)
# n dels, n-1 trans, 26n reps, and 26(n+1) ins, totally 54n+25

def known_edits2(word):
    return set(e2 for e1 in edits1(word) for e2 in edits1(e1) if e2 in NWORDS)
# only consider those are known in the training set
```

Solving the Problem

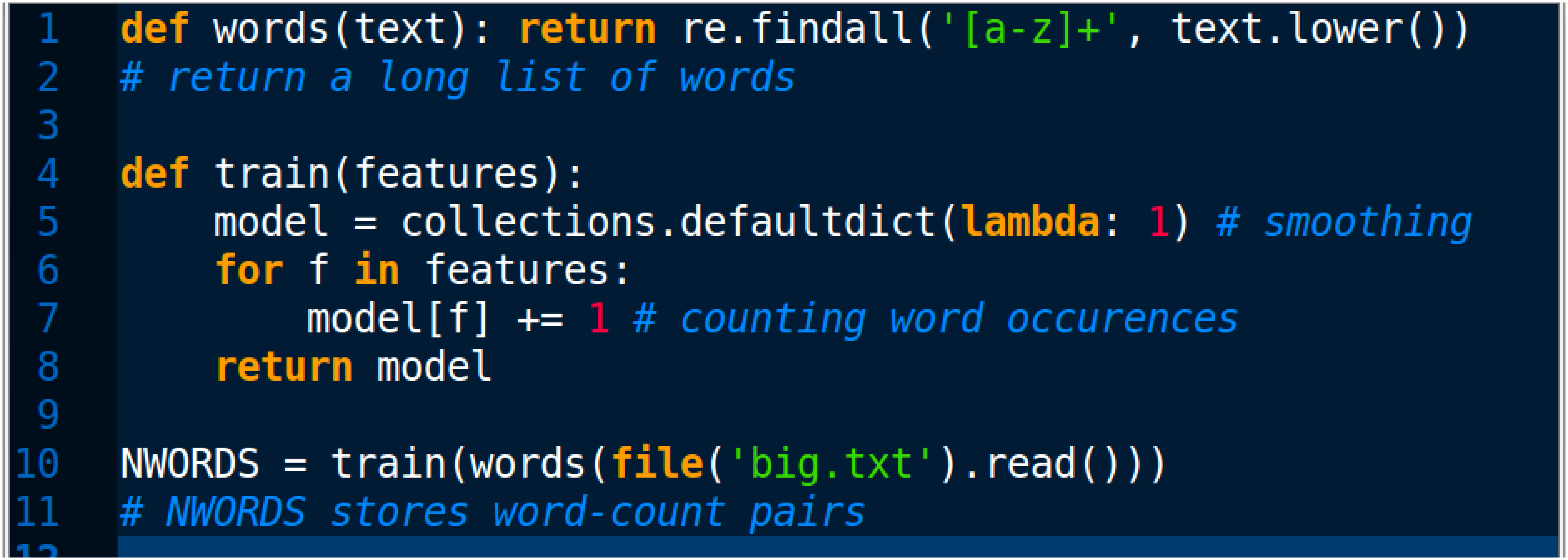
- argmax_c P(w|c) P(c)
  - argmax_c
    - cover all possible corrections ideally
  - P(c)
    - Language Model
    - capturing how likely **c** would stand on its own
  - P(w|c)
    - Error model
    - capturing how like the change **c** => **w** will happen

Building Language Model P(c)

- Training data
  - counts of each word in the training data
  - smoothing for novel words

Building Language Model P(c)

- Train the languge model



```
def words(text): return re.findall('[a-z]+', text.lower())
# return a long list of words

def train(features):
    model = collections.defaultdict(lambda: 1) # smoothing
    for f in features:
        model[f] += 1 # counting word occurences
    return model

NWORDS = train(words(file('big.txt').read()))
# NWORDS stores word-count pairs
```

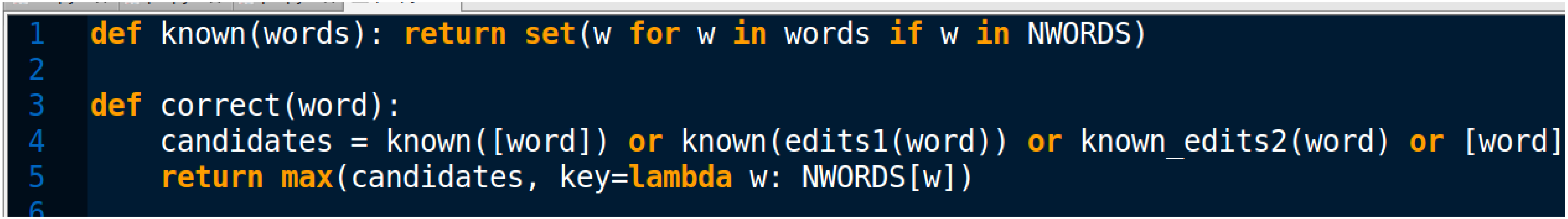
Solving the Problem

- argmax_c P(w|c) P(c)
  - argmax_c
    - cover all possible corrections ideally
  - P(c)
    - Language Model
    - capturing how likely **c** would stand on its own
  - P(w|c)
    - Error model
    - capturing how like the change **c** => **w** will happen

Building Error Model P(w|c)

- Training data
- Trivial model

P(any known edit0 word) >> P(any known edit1 word) >> P(any known edit2 word)



```
def known(words): return set(w for w in words if w in NWORDS)

def correct(word):
    candidates = known([word]) or known(edits1(word)) or known_edits2(word) or [word]
    return max(candidates, key=lambda w: NWORDS[w])
```

Solving the Problem

- argmax_c P(w|c) P(c)
  - argmax_c
    - cover all possible corrections ideally
  - P(c)
    - Language Model
    - capturing how likely **c** would stand on its own
  - P(w|c)
    - Error model
    - capturing how like the change **c** => **w** will happen

Evaluation

-
  ```
  >>> correct('speling')
  'spelling'
  >>> correct('korrecter')
  'corrector'
  ```

- Testing data
  - Roger Mitton's Birkbeck spelling error corpus
  - ~70% correctness on two datasets

Improvement

- Places can we improve
  - argmax_c P(w|c) P(c)
    - argmax_c
    - P(c)
    - P(w|c)

Improvement

- Improving coverage of possible corrections (argmax_c)
  - words beyond edit distance 2
    - Examples:
      - successful sucssuful
      - hierarchy heiarky
      - profession preffeson

  - allowing a limited set of edits at edit distance 3

Improvement

- Improving the languge model P(c)
  - deal with unknown words
    - bad case1: correct('economtric') => 'economic' (121); expected 'econometric' (1)
    - bad case 2: correct('generataed') => 'generate' (2); expected 'generated' (1)

    - Possible solutions
      - more training data
      - add different forms of a word, e.g. -s to a noun, -ed to a verb
      - sequences of characters, e.g. suffix "-ally"

Improvement

- Improving the error model P(w|c)
  - P(any known edit_0 word) >> P(any known edit1 word) >> P(any known edit2 word)
  - bad cases:
    - correct('reciet') => 'recite' (5); expected 'receipt' (14)
    - correct('thay') => 'that' (12513); expected 'they' (4939)
    - correct('wonted') => 'wonted' (2); expected 'wanted' (214)
  - possible solution
    - alternative similarity metrics
    - a corpus of spelling errors

Improvement

- Beyond the three major factors
  - Context
    - examples
      - correct('carrers') => 'carriers' (7); expected 'careers' (2)
      - correct('quies') => 'quiet' (119); expected 'queries' (1)
    - Google n-gram will be very useful

Improvement

- Beyond the three major factors
  - Fixing errors in testing data
    - examples
      - correct('sumarys') => 'summary' (17); expected 'summarys' (1)
      - correct('humor') => 'humor' (17); expected 'humour' (5)
  - Improve response time
    - results caching

Useful Links

- Original tutorial “How to write a spelling corrector”:

  http://norvig.com/spell-correct.html

- Other versions:

  Perl: http://www.riffraff.info/2007/5/20/a-spell-corrector-in-perl6-part-3
  Java: http://raelcunha.com/spell-correct.php
  C:http://scarvenger.wordpress.com/2007/12/11/how-to-write-a-spelling-corrector/

- aspell project:

  http://aspell.net/test/ (more testing data)

- Toolkit: LingPipe:

  http://alias-i.com/lingpipe/demos/tutorial/querySpellChecker/read-me.html