Como Escribir un Corrector Ortografico en 20 Lineas

Habla: Roberto Alsina <ralsina@netmanagers.com.ar>

Codigo: Norvig & Bacon

El Codigo

```
import re, collections

def words(text): return re.findall('[a-z]+', text.lower())

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

NWORDS = train(words(file('big.txt').read()))

alphabet = 'abcdefghijklmnopqrstuvwxyz'

def edits1(word):
   s = [(word[:i], word[i:]) for i in range(len(word) + 1)]
   deletes    = [a + b[1:] for a, b in s if b]
   transposes = [a + b[1] + b[0] + b[2:] for a, b in s if len(b)>1]
   replaces   = [a + c + b[1:] for a, b in s for c in alphabet if b]
   inserts    = [a + c + b     for a, b in s for c in alphabet]
```

El Codigo

```
   return set(deletes + transposes + replaces + inserts)

def known_edits2(word):
    return set(e2 for e1 in edits1(word) for e2 in edits1(e1) if e2 in NWORDS)

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=NWORDS.get)
```

Como funciona (La teoria)

Todo esto es culpa del teorema de Bayes (ignorenme!).

Queremos saber en que palabra estaba pensando el usuario. Entonces:

- Cual es la probabilidad de que sea exactamente lo que escribio? P(w)
- Dado un reemplazo c, cual es la probabilidad que sea esa? P(c)
- Cual es la probabilidad de que haya puesto w queriendo poner c? P(w|c)
- En realidad, queremos maximizar P(w|c) * P(c)

  O sea: queremos la palabra c mas probable que se parezca mas a w.

  - Si es muy distinta de w, c no nos sirve (w=berde, c=hola)
  - Si es muy rara, no nos sirve (w=verdo, c=bardo)
  - Si es comun y parecida, c es genial!

Codigo!!!!!

POR FAVOR!

Calculando P(c)

- Agarra un montooooooon de texto y conta las palabras.
- Indexalo por palabra
- Listo

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

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

NWORDS = train(words(file('big.txt').read()))
```

Calculando P(w|c)

Que quiere decir parecida?

- Distancia de edicion

  Cuantas veces tengo que editar una palabra para convertirla en otra.
  - hola -> bola = 1 (alteracion)
  - hola -> ola = 1 (eliminacion)
  - hola -> holas = 1 (insercion)
  - hola -> ohla = 1 (transposicion)

CODIGO!!!!

Aca viene...

Esto te da el conjunto de todas las palabrs a distancia 1 de otra:

```
def edits1(word):
    n = len(word)
    return set([word[0:i]+word[i+1:] for i in range(n)] +                     # deletion
               [word[0:i]+word[i+1]+word[i]+word[i+2:] for i in range(n-1)] + # transposition
               [word[0:i]+c+word[i+1:] for i in range(n) for c in alphabet] + # alteration
               [word[0:i]+c+word[i:] for i in range(n+1) for c in alphabet])  # insertion
```

El 80% de los errores son a distancia 1. Si queres distancia 2 (95% de los errores), es usar edits1 dos veces ;-)

```
def edits2(word):
    return set(e2 for e1 in edits1(word) for e2 in edits1(e1))
```

SON DEMASIADAS!

En realidad, edits2 da MUCHO. Entonces descartamos todo lo que no puede ser una correccion porque nunca lo vimos escrito en un texto (tiene P(c)=0)

```
def known_edits2(word):
    return set(e2 for e1 in edits1(word) for e2 in edits1(e1) if e2 in NWORDS)

len(known_edits2('something'))

3

len(edits2('something')

114324
```

ESTIMANDO P(w|c)

```
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])
```

- known descarta las palabras que nunca vimos en uso.
- correct dice:
  - si word es conocida, la correccion es word. Si no...
  - si hay palabras conocidas a distancia 1, es la mas probable de esas. Si no...
  - si hay palabras conocidas a distancia 2, es la mas probable de esas. Si no....
  - me rindo

Y funciona????

Si:

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