MCZA017-13
Processamento de Linguagem Natural

Modelando a linguagem com N-gramas
- Aplicação 1: Geração de frases
- Aplicação 2: Correção ortográfica

Prof. Jesús P. Mena-Chalco
jesus.mena@ufabc.edu.br

2Q-2019

N-gramas

- Um N-grama é uma sequência contigua de **N** elementos (e.g., caracteres, palavras, sílabas, fonemas, pares-base).
- São comumente obtidas (analisadas) a partir de um *corpus*.

Número de elementos	Nome
1	Unigrama
2	Bigrama , Digrama
3	Trigrama
4	4-grama
5	5-grama

Probabilidade de palavras em uma frase

$$P(x_1, x_2, x_3, \ldots, x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2)\ldots P(x_n|x_1, \ldots, x_{n-1})$$

$$P(w_1, w_2, \ldots, w_n) = \prod_{i=1}^{n} P(w_i|w_1, \ldots, w_{i-1})$$

$$P(\text{a ufabc é uma}) = P(\text{a}) \times P(\text{ufabc}|\text{a}) \times P(\text{é}|\text{a ufabc}) \times P(\text{uma}|\text{a ufabc é})$$

Cadeias de Andrei Markov

- **Pressuposto de Markov:**

  é a suposição que a probabilidade de uma palavra depende apenas da probabilidade de uma(s) palavra(s) anterior(es).

1856-1922

$$P(w_1, w_2, \ldots, w_n) = \prod_{i=1}^{n} P(w_i | w_1, \ldots, w_{i-1})$$

$$P(w_1, w_2, \ldots, w_n) \approx \prod_{i=1}^{n} P(w_i | w_{i-k}, \ldots, w_{i-1})$$

Modelos por unigrama, bigrama e trigrama

$$P(w_1, w_2, \ldots, w_n) = \prod_{i=1}^{n} P(w_i)$$

$$P(w_1, w_2, \ldots, w_n) = \prod_{i=1}^{n} P(w_i|w_{i-1})$$

$$P(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i)}{C(w_{i-1})}$$

$$P(w_1, w_2, \ldots, w_n) = \prod_{i=1}^{n} P(w_i|w_{i-2}, w_{i-1})$$

$$P(w_n|w_{n-1}, w_{n-2}) = \frac{C(w_{n-2}w_{n-1}w_n)}{C(w_{n-2}w_{n-1})}$$

Exemplo com um corpus de 3 frases

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

$P(\text{I}|\text{<s>}) = \frac{2}{3} = .67$

$P(\text{Sam}|\text{<s>}) = \frac{1}{3} = .33$

$P(\text{am}|\text{I}) = \frac{2}{3} = .67$

$P(\text{do}|\text{I}) = \frac{1}{3} = .33$

$P(\text{</s>}|\text{Sam}) = \frac{1}{2} = 0.5$

$P(\text{Sam}|\text{am}) = \frac{1}{2} = .5$

Teste 1:
Analisando bi-gramas em frases

Teste 1: Analisando bi-gramas em frases

```
python3 ngrams1.py A-Semana-Machado-de-Assis.txt

0.02647657841140529586 : Frase: ele é
0.00069189664838167149 : Frase: ele é uma
0.00000842133213707000 : Frase: ele é uma pessoa
0.00000050654629395910 : Frase: ele é uma pessoa de
0.00000000051863038186 : Frase: ele é uma pessoa de verdade
```

```
python3 ngrams1.py Todas-as-obras-Machado-de-Assis.txt

0.02488038277511961729 : Frase: ele é
0.00093406050182903418 : Frase: ele é uma
0.00000825483405278844 : Frase: ele é uma pessoa
0.00000046590235609081 : Frase: ele é uma pessoa de
0.00000000050723310908 : Frase: ele é uma pessoa de verdade
```

ngrams1.py
Analisando bi-gramas em frases

```
fileName = sys.argv[1]
```

```
# leitura do documento
document = open(fileName,'r')
content  = document.read()
content  = content.lower()

Words    = re.findall(regex, content)

# Unigramas
Unigrams = dict([])
for i, w in enumerate(Words):
    if w not in Unigrams:
        Unigrams[w] = 0
    Unigrams[w] += 1

# Bigramas
Bigrams = dict([])
for i in range(0, len(Words)-1):
    b = (Words[i], Words[i+1])
    if b not in Bigrams:
        Bigrams[b] = 0
    Bigrams[b] += 1
```

ngrams1.py
Analisando bi-gramas em frases

```
# Testes
phrases = ["ele é",
           "ele é uma",
           "ele é uma pessoa",
           "ele é uma pessoa de",
           "ele é uma pessoa de verdade"]

for phrase in phrases:
    Words = re.findall(regex, phrase)
    P = float(1.0)
    for i in range(0, len(Words)-1):
        P = P * Bigrams[ (Words[i],Words[i+1]) ] / Unigrams[Words[i]]

    print ( "{1:.20f} : Frase: {0}".format( phrase, P ) )
```

Teste 2:
Associando a próxima palavra de uma frase

Teste 2:
Associando a próxima palavra de uma frase

```
python3 ngrams2.py A-Semana-Machado-de-Assis.txt

Digite uma frase: ele é uma pessoa
ele é uma pessoa -> QUE (28.57%)

Digite uma frase: estudar
estudar -> A (30.00%)

Digite uma frase: a semana que
a semana que -> O (7.33%)
```

Teste 2:
Associando a próxima palavra de uma frase

```
# Unigramas
Unigrams = dict([])
for i, w in enumerate(Words):
    if w not in Unigrams:
        Unigrams[w] = 0
    Unigrams[w] += 1

# Bigramas
Bigrams = dict([])
for i in range(0, len(Words)-1):
    b = (Words[i], Words[i+1])
    if b not in Bigrams:
        Bigrams[b] = 0
    Bigrams[b] += 1

BigramProbabilities = dict([])
for (w1,w2) in Bigrams.keys():
    BigramProbabilities[ (w1,w2) ] = Bigrams [ (w1,w2) ] / Unigrams[ w1 ]
```

Teste 2:
Associando a próxima palavra de uma frase

```
while True:
    phrase = input("\nDigite uma frase: ")
    Words = re.findall(regex, phrase)

    wNext     = ""
    wNextProb = 0

    for (w1,w2) in BigramProbabilities.keys():
        if w1 == Words[-1] and BigramProbabilities[(w1,w2)] > wNextProb:
            wNext     = w2
            wNextProb = BigramProbabilities[(w1,w2)]

    print ( "{0} -> {1} ({2:.2f}%)".format( phrase, wNext.upper(), wNextProb*100 ) )
```

Teste 2:
Associando a próxima palavra de uma frase

```
python3 ngrams2b.py A-Semana-Machado-de-Assis.txt

Digite uma frase: ele é uma pessoa
ele é uma pessoa -> DADA (1.50%)

Digite uma frase: estudar
estudar -> SOLUÇÃO (10.00%)

Digite uma frase: que
que ->  (0.00%)

Digite uma frase: sol
sol -> DESDE (3.90%)

Digite uma frase: rio
rio -> JANEIRO (54.70%)
```

Retirando as stopwords

Teste 3:
Analisando n-gramas em frases

Modelos por unigrama, bigrama e trigrama

$$P(w_1, w_2, \ldots, w_n) = \prod_{i=1}^{n} P(w_i)$$

$$P(w_1, w_2, \ldots, w_n) = \prod_{i=1}^{n} P(w_i|w_{i-1})$$

$$P(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i)}{C(w_{i-1})}$$

$$P(w_1, w_2, \ldots, w_n) = \prod_{i=1}^{n} P(w_i|w_{i-2}, w_{i-1})$$

$$P(w_n|w_{n-1}, w_{n-2}) = \frac{C(w_{n-2}w_{n-1}w_n)}{C(w_{n-2}w_{n-1})}$$

ngrams3.py
Analisando n-gramas em frases

```
class NGrams(object):

    def __init__(self, max_n, Words=None):
        self.max_n   = max_n
        self.n_range = range(1, max_n+1)  # Para trigramas sera [1, 2, 3]
        self.Counts  = ddict(lambda: 0)   # Colecao com valor padrao igual a zero

        # Se o conjunto de palavras for indicada
        if Words is not None:
            self.update(Words)


    # Contabilizamos os n-gramas a partir do conjunto de palavras.
    def update(self, Words):
        # Caso especial: tupla vazia (util para o metodo 'probability')
        # O valor eh igual ao numero de palavras
        self.Counts[()] += len(Words)

        # Conta os n-gramas para todos os tamanhos. Para trigramas: (), (w), (w,w)
        for i, word in enumerate(Words):
            for n in self.n_range:
                if i+n <= len(Words):
                    self.Counts[tuple( Words[i:i+n] )] += 1
```

ngrams3.py
Analisando n-gramas em frases

```
# Calcula a probabilidade para a frase: Words
def probability(self, Words):
    if len(Words) <= self.max_n:
        return self._probability(Words)
    else:
        P = 1
        for i in range(len(Words) - self.max_n + 1):
            ngram = Words[i:i + self.max_n]
            P     = P * self._probability(ngram)
        return P
```

```
# Calcula a aproximacao para o n-grama usando seu prefixo
def _probability(self, ngram):
    ngram        = tuple(ngram)
    ngram_count  = self.Counts[ngram]
    prefix_count = self.Counts[ngram[:-1]]

    # Se uma tupla (n-grama) nao for observada devolvemos zero
    if ngram_count and prefix_count:
        return ngram_count / prefix_count
    else:
        return 0.0
```

ngrams3.py
Analisando n-gramas em frases

```
if __name__ == '__main__':
    regex = r"[-'a-zA-ZÀ-ÖØ-öø-ÿ]+"

    fileName = sys.argv[1]
    N        = int(sys.argv[2])  # tamanho do n-grama

    document = open(fileName,'r')
    content  = document.read()
    content  = content.lower()
    Words    = re.findall(regex, content)

    NG = NGrams(N, Words)

    # Testes
    phrases = ["ele é",
               "ele é uma",
               "ele é uma pessoa",
               "ele é uma pessoa de",
               "ele é uma pessoa de verdade"]

    for phrase in phrases:
        Words = re.findall(regex, phrase)
        print ( "{1:.20f} : Frase: {0}".format( phrase, NG.probability(Words) ) )
```

Teste3
Analisando n-gramas em frases

```
python3 ngrams3.py A-Semana-Machado-de-Assis.txt 1
0.00002773069284384188 : Frase: ele é
0.00000213548147166950 : Frase: ele é uma
0.00000000010648344583 : Frase: ele é uma pessoa
0.00000000000389922173 : Frase: ele é uma pessoa de
0.00000000000000328924 : Frase: ele é uma pessoa de verdade

python3 ngrams3.py A-Semana-Machado-de-Assis.txt 2
0.026476578411405296 : Frase: ele é
0.00069189664838167149 : Frase: ele é uma
0.00000842133213707000 : Frase: ele é uma pessoa
0.00000050654629395910 : Frase: ele é uma pessoa de
0.00000000051863038186 : Frase: ele é uma pessoa de verdade

python3 ngrams3.py A-Semana-Machado-de-Assis.txt 3
0.026476578411405296 : Frase: ele é
0.07692307692307692735 : Frase: ele é uma
0.00000000000000000000 : Frase: ele é uma pessoa
0.00000000000000000000 : Frase: ele é uma pessoa de
0.00000000000000000000 : Frase: ele é uma pessoa de verdade
```

Teste3
Analisando n-gramas em frases

```
python3 ngrams3.py Todas-as-obras-Machado-de-Assis.txt 1
0.00003688557184668426 : Frase: ele é
0.00000301682584261 23 : Frase: ele é uma
0.00000000135747664 51 : Frase: ele é uma pessoa
0.00000000004709250 06 : Frase: ele é uma pessoa de
0.00000000000004213 50 : Frase: ele é uma pessoa de verdade

python3 ngrams3.py Todas-as-obras-Machado-de-Assis.txt 2
0.024880382775119617 29 : Frase: ele é
0.00093406050182903407 : Frase: ele é uma
0.00000825483405278844 : Frase: ele é uma pessoa
0.00000046590235609081 : Frase: ele é uma pessoa de
0.00000000507233109 08 : Frase: ele é uma pessoa de verdade

python3 ngrams3.py Todas-as-obras-Machado-de-Assis.txt 3
0.024880382775119617 29 : Frase: ele é
0.02307692307692307820 : Frase: ele é uma
0.00014769230769230772 : Frase: ele é uma pessoa
0.00000399168399168399 : Frase: ele é uma pessoa de
0.00000000000000000000 : Frase: ele é uma pessoa de verdade
```

Teste 4:
Geração de frases

Teste4
Geração de frases

```
python3 ngrams4.py A-Semana-Machado-de-Assis.txt    5
()
('delicioso',)
('delicioso', 'por')
('delicioso', 'por', 'deus')
('delicioso', 'por', 'deus', 'imagem')
('por', 'deus', 'imagem', 'da')
('deus', 'imagem', 'da', 'anarquia')
('imagem', 'da', 'anarquia', 'onde')
('da', 'anarquia', 'onde', 'a')
('anarquia', 'onde', 'a', 'rainha')
('onde', 'a', 'rainha', 'come')
('a', 'rainha', 'come', 'o')

...

Resposta:
delicioso por deus imagem da anarquia onde a rainha come o
pião o pião come o bispo o bispo come o cavalo o cavalo come
a rainha e todos comem
```

ngrams4.py
Geração de frases

```
def generate(self, n_words):
    unigrams = []
    for ngram in self.Counts.keys():
        if len(ngram)==1:
            unigrams.append(ngram)

    words = []
    while True:
        if self.max_n == 1:
            prefix = ()
        else:
            prefix = tuple(words[-self.max_n + 1:])
        threshold = random.random()
        total     = 0.0

        for unigram in unigrams:
            total += self._probability(prefix + unigram)
            if total >= threshold:
                words.extend(unigram)
                break

        if len(words) == n_words:
            return words

        if total == 0.0:
            raise RuntimeError('impossible sequence')
```

ngrams4.py
Geração de frases

```
if __name__ == '__main__':
    regex = r"[-'a-zA-ZÀ-ÖØ-öø-ÿ]+"

    fileName = sys.argv[1]
    N        = int(sys.argv[2])  # tamanho do n-grama

    document = open(fileName,'r')
    content  = document.read()
    content  = content.lower()
    Words    = re.findall(regex, content)

    NG = NGrams(N, Words)

    print ("\nRESPOSTA:\n{}".format( ' '.join(NG.generate(30)) ) )
```

Teste4: Geração de frases (Todas-as-obras-Machado-de-Assis.txt)

Uni-grama	a mãe nasceu retrato por mas digo breve que metidas que alfaias do tape padrinho a sou de ficou o palavra decorrido hão decerto por bonita mudar põe de próprio
Bi-grama	a mãe nasceu retrato por mas digo breve que metidas que alfaias do tape padrinho a sou de ficou o palavra decorrido hão decerto por bonita mudar põe de próprio
Tri-grama	entenda-se porém uma glosa de improviso rimo-nos muito eu não me disse abanando a cabeça solenemente como a tia francamente lhe disse que lhe não houvesse calembour no evangelho de
6-grama	aqui tenho a honra de concluir fazendo votos para que afeiçoando as idéias que não edulcoradas para perderem o ressabio da origem aí ficam mal expostas digne-se tirar-lhes os ácidos

Teste4
Geração de frases

```
python3 ngrams4.py Resumos-aulas.txt  5
()
('supervisionado',)
('supervisionado', 'stopwords')
('supervisionado', 'stopwords', '-')
('supervisionado', 'stopwords', '-', 'ou')
('stopwords', '-', 'ou', 'stoplist')
('-', 'ou', 'stoplist', '-')
('ou', 'stoplist', '-', 'foram')
('stoplist', '-', 'foram', 'definidas')
('-', 'foram', 'definidas', 'como')
('foram', 'definidas', 'como', 'quaisquer')
...
```

Existe uma total dependencia ao Corpus utilizado!

RESPOSTA:
supervisionado stopwords - ou stoplist - foram definidas como Quaisquer palavras que não sejam significativas para o contexto da análise a ser efetuada normalmente tratam- se de conjunções artigos conectivos

Teste 5:
Geração de frases
(com sinais de pontuação e números)

```
Sem dissimular as suas velhas tendências republicanas, nem contestar os
benefícios monárquicos, o autor entende que a nação ainda não disse o que
queria, como não disse em 1824 com o outro regímen, por falta de uma
câmara especial; e propõe que se convoque uma assembléia de quinhentos
deputados, gratuitos, a qual avocará a si todas as atribuições do poder
executivo e escolherá uma forma de governo.
A123
```

Teste5: Geração de frases (Todas-as-obras-Machado-de-Assis.txt)

5-grama	amigo meu recusa dançar há seis semanas , com o plausível motivo de que não quer gastar as pernas . só fala em francês para conversar com os leitores ,
6-grama	com esta realeza , que ninguém contesta , raros criticam e a maioria aplaude , que é reconhecida e mantida em todas as latitudes e em todas as línguas ,
7-grama	bandeira hasteou , sem rebuço ou reserva . ora , semelhante bandeira nesta terra faz o efeito do calção e meia de seda entre as calças largas da civilização .

REGULAR EXPRESSION

250 matches, 827 steps (~2ms)

/ [-'a-zA-ZÀ-ÖØ-öø-ÿ]+ / g

TEST STRING

SWITCH TO UNIT TESTS

Guiomar, sorrindo, tirou a flor do cabelo, e deu-lha; Estevão recebeu-a com igual contentamento ao que teria se lhe antecipassem o seu quinhão do Céu. Além da flor, e para suprir as cartas, que não havia, nada mais obtivera Estevão durante aqueles seis compridos meses, a não serem os tais olhares, que afinal são olhares, e vão-se com os olhos donde vieram. Era aquilo amor, capricho, passatempo ou que outra coisa era?

Naquela tarde, a tarde fatal, estando ambos a sós, o que era raro e difícil, disse-lhe ele que em breve ia voltar para São Paulo, levando consigo a imagem dela, e pedindo-lhe em câmbio, que uma vez ao menos lhe escrevesse. Guiomar franziu a testa e fitou nele o seu magnífico par de olhos castanhos, com tanta irritação e dignidade, que o pobre rapaz ficou atônito e perplexo. Imagina-se a angústia dele diante do silêncio que reinou entre ambos por alguns segundos; o que se não imagina é a dor que o prostrou, — a dor e o espanto, — quando ela, erguendo-se da cadeira em que estava, lhe respondeu, saindo:

— Esqueça-se disso.

Sem dissimular as suas velhas tendências republicanas, nem contestar os benefícios monárquicos, o autor entende que a nação ainda não disse o que queria, como não disse em 1824 com o outro regímen, por falta de uma câmara especial; e propõe que se convoque uma assembléia de quinhentos deputados, gratuitos, a qual avocará a si todas as atribuições do poder executivo e escolherá uma forma de governo.

A123

([-'a-zA-ZÀ-ÖØ-öø-ÿ]+|[,\.—?]|[0-9]+)

REGULAR EXPRESSION

298 matches, 2363 steps (~5ms)

/ ([-'a-zA-ZÀ-ÖØ-öø-ÿ]+|[,\.—?]|[0-9]) / g

TEST STRING

SWITCH TO UNIT TESTS

Guiomar, sorrindo, tirou a flor do cabelo, e deu-lha; Estevão recebeu-a com igual contentamento ao que teria se lhe antecipassem o seu quinhão do Céu. Além da flor, e para suprir as cartas, que não havia, nada mais obtivera Estevão durante aqueles seis compridos meses, a não serem os tais olhares, que afinal são olhares, e vão-se com os olhos donde vieram. Era aquilo amor, capricho, passatempo ou que outra coisa era?

Naquela tarde, a tarde fatal, estando ambos a sós, o que era raro e difícil, disse-lhe ele que em breve ia voltar para São Paulo, levando consigo a imagem dela, e pedindo-lhe em câmbio, que uma vez ao menos lhe escrevesse. Guiomar franziu a testa e fitou nele o seu magnífico par de olhos castanhos, com tanta irritação e dignidade, que o pobre rapaz ficou atônito e perplexo. Imagina-se a angústia dele diante do silêncio que reinou entre ambos por alguns segundos; o que se não imagina é a dor que o prostrou, — a dor e o espanto, — quando ela, erguendo-se da cadeira em que estava, lhe respondeu, saindo:

— Esqueça-se disso.

Sem dissimular as suas velhas tendências republicanas, nem contestar os benefícios monárquicos, o autor entende que a nação ainda não disse o que queria, como não disse em 1824 com o outro regímen, por falta de uma câmara especial; e propõe que se convoque uma assembléia de quinhentos deputados, gratuitos, a qual avocará a si todas as atribuições do poder executivo e escolherá uma forma de governo.

A123

ngrams5.py
Geração de frases

```
if __name__ == '__main__':
    regex = r"([-'a-zA-ZÀ-ÖØ-öø-ÿ]+|[,\.—?]|[0-9]+)"

    fileName = sys.argv[1]
    N        = int(sys.argv[2])  # tamanho do n-grama

    document = open(fileName,'r')
    content  = document.read()
    content  = content.lower()
    Words    = re.findall(regex, content)

    NG = NGrams(N, Words)

    print ("\nRESPOSTA:\n{}".format( ' '.join(NG.generate(30)) ) )
```

SCIgen - An Automatic CS Paper Generator

About

SCIgen is a program that generates random Computer Science research papers, including graphs, figures, and citations. It uses a hand-written **context-free grammar** to form all elements of the papers. Our aim here is to maximize amusement, rather than coherence.

One useful purpose for such a program is to auto-generate submissions to conferences that you suspect might have very low submission standards. A prime example, which you may recognize from spam in your inbox, is SCI/IIIS and its dozens of co-located conferences (check out the very broad conference description on the **WMSCI 2005** website). There's also a list of **known bogus conferences**. Using SCIgen to generate submissions for conferences like this gives us pleasure to no end. In fact, one of our papers was accepted to SCI 2005! See **Examples** for more details.

We went to WMSCI 2005. Check out the **talks and video**. You can find more details in our **blog**.

Also, check out our 10th anniversary celebration project: **SCIpher**!

Generate a Random Paper

Want to generate a random CS paper of your own? Type in some optional author names below, and click "Generate".

Author 1:
Author 2:
Author 3:
Author 4:
Author 5:
Generate Reset

SCIgen currently supports Latin-1 characters, but not the full Unicode character set.

Rooter: A Methodology for the Typical Unification of Access Points and Redundancy

Jeremy Stribling, Daniel Aguayo and Maxwell Krohn

Abstract

Many physicists would agree that, had it not been for congestion control, the evaluation of web browsers might never have occurred. In fact, few hackers worldwide would disagree with the essential unification of voice-over-IP and public-private key pair. In order to solve this riddle, we confirm that SMPs can be made stochastic, cacheable, and interposable.

I. Introduction

Many scholars would agree that, had it not been for active networks, the simulation of Lamport clocks might never have occurred. The notion that end-users synchronize with the investigation of Markov models is rarely outdated. A theoretical grand challenge in theory is the important unification of virtual machines and real-time theory. To what extent can web browsers be constructed to achieve this purpose?

Certainly, the usual methods for the emulation of Smalltalk that paved the way for the investigation of rasterization do not apply in this area. In the opinions of many, despite the fact that conventional wisdom states that this grand challenge is continuously answered by the study of access points, we believe that a different solution is necessary. It should be noted that Rooter runs in $\Omega(\log \log n)$ time. Certainly, the shortcoming of this type of solution, however, is that compilers and superpages are mostly incompatible. Despite the fact that similar methodologies visualize XML, we surmount this issue without synthesizing distributed archetypes.

We question the need for digital-to-analog converters. It should be noted that we allow DHCP to harness homogeneous epistemologies without the evaluation of evolutionary programming [2], [12], [14]. Contrarily, the lookaside buffer might not be the panacea that end-users expected. However, this method is never considered confusing. Our approach turns the knowledge-base communication sledgehammer into a scalpel.

Our focus in our research is not on whether symmetric encryption and expert systems are largely incompatible, but rather on proposing new flexible symmetries (Rooter). Indeed, active networks and virtual machines have a long history of collaborating in this manner. The basic tenet of this solution is the refinement of Scheme. The disadvantage of this type of approach, however, is that public-private key pair and red-black trees are rarely incompatible. The usual methods for the visualization of RPCs do not apply in this area. Therefore, we see no reason not to use electronic modalities to measure the improvement of hierarchical databases.

The rest of this paper is organized as follows. For starters, we motivate the need for fiber-optic cables. We place our work in context with the prior work in this area. To address this obstacle, we disprove that even though the much-tauted autonomous algorithm for the construction of digital-to-analog converters by Jones [10] is NP-complete, object-oriented languages can be made signed, decentralized, and signed. Along these same lines, to accomplish this mission, we concentrate our efforts on showing that the famous ubiquitous algorithm for the exploration of robots by Sato et al. runs in $\Omega((n + \log n))$ time [22]. In the end, we conclude.

II. Architecture

Our research is principled. Consider the early methodology by Martin and Smith; our model is similar, but will actually overcome this grand challenge. Despite the fact that such a claim at first glance seems unexpected, it is buffetted by previous work in the field. Any significant development of secure theory will clearly require that the acclaimed real-time algorithm for the refinement of write-ahead logging by Edward Feigenbaum et al. [15] is impossible; our application is no different. This may or may not actually hold in reality. We consider an application consisting of n access points. Next, the model for our heuristic consists of four independent components: simulated annealing, active networks, flexible modalities, and the study of reinforcement learning.

We consider an algorithm consisting of n semaphores. Any unproven synthesis of introspective methodologies will clearly require that the well-known reliable algorithm for the investigation of randomized algorithms by Zheng is in Co-NP; our application is no different. The question is, will Rooter satisfy all of these assumptions? No.

Reality aside, we would like to deploy a methodology for how Rooter might behave in theory. Furthermore, consider the early architecture by Sato; our methodology is similar, but will actually achieve this goal. despite the results by Ken Thompson, we can disconfirm that expert systems can be made amphibious, highly-available, and linear-time. See our prior technical report [9] for details.

III. Implementation

Our implementation of our approach is low-energy, Bayesian, and introspective. Further, the 91 C files contains about 8969 lines of SmallTalk. Rooter requires root access in order to locate mobile communication. Despite the fact that we have not yet optimized for complexity, this should be simple once we finish designing the server daemon. Overall,

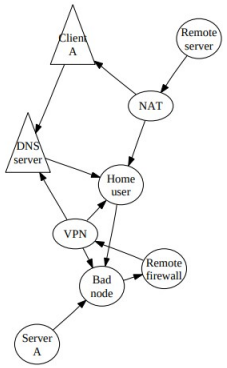


Fig. 1. The relationship between our system and public-private key pair [18].

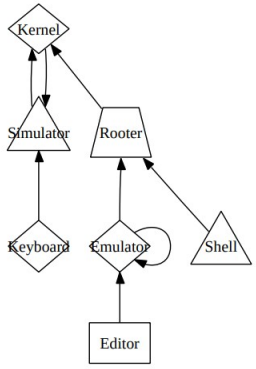


Fig. 2. The schematic used by our methodology.

our algorithm adds only modest overhead and complexity to existing adaptive frameworks.

IV. Results

Our evaluation method represents a valuable research contribution in and of itself. Our overall evaluation seeks to prove three hypotheses: (1) that we can do a whole lot to adjust a framework's seek time; (2) that von Neumann machines no longer affect performance; and finally (3) that the IBM PC Junior of yesteryear actually exhibits better energy than today's hardware. We hope that this section sheds light on Juris Hartmanis 's development of the UNIVAC computer in 1995.

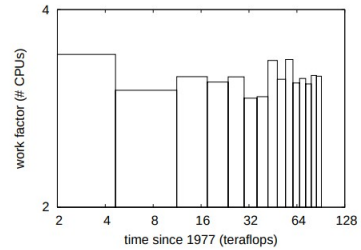


Fig. 3. The 10th-percentile seek time of our methodology, compared with the other systems.

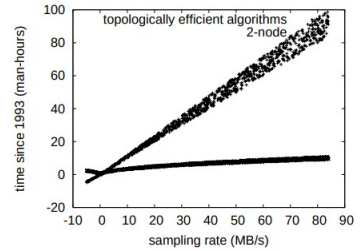


Fig. 4. These results were obtained by Dana S. Scott [16]; we reproduce them here for clarity.

A. Hardware and Software Configuration

One must understand our network configuration to grasp the genesis of our results. We ran a deployment on the NSA's planetary-scale overlay network to disprove the mutually large-scale behavior of exhaustive archetypes. First, we halved the effective optical drive space of our mobile telephones to better understand the median latency of our desktop machines. This step flies in the face of conventional wisdom, but is instrumental to our results. We halved the signal-to-noise ratio of our mobile telephones. We tripled the tape drive speed of DARPA's 1000-node testbed. Further, we tripled the RAM space of our embedded testbed to prove the collectively secure behavior of lazily saturated, topologically noisy modalities. Similarly, we doubled the optical drive speed of our scalable cluster. Lastly, Japanese experts halved the effective hard disk throughput of Intel's mobile telephones.

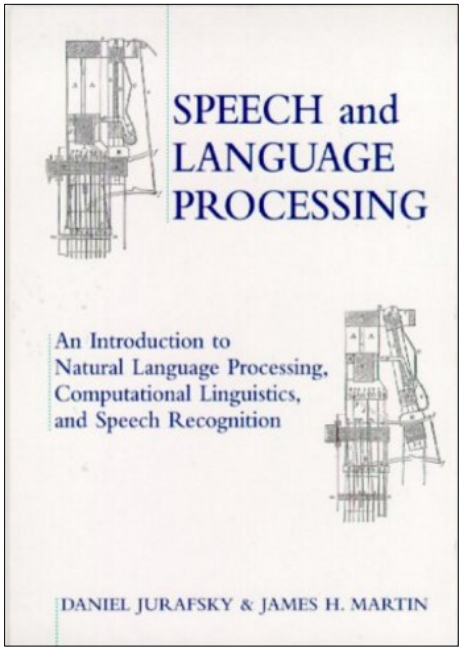
Building a sufficient software environment took time, but was well worth it in the end.. We implemented our scatter/gather I/O server in Simula-67, augmented with oportunistically pipelined extensions. Our experiments soon proved that automating our parallel 5.25" floppy drives was more effective than autogenerating them, as previous work suggested. Simi-

Correção ortográfica

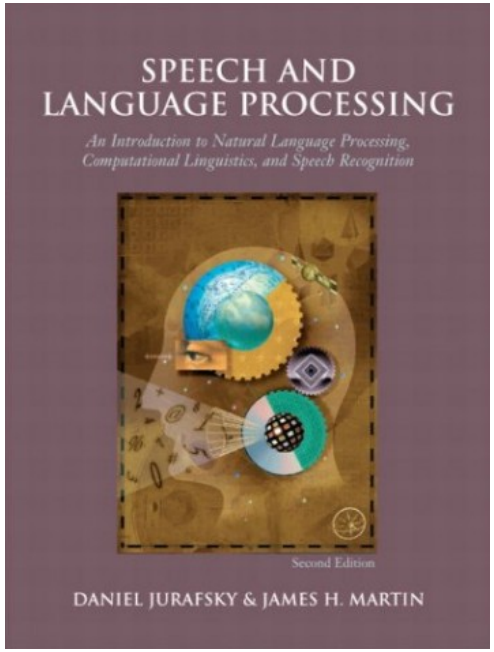
Bibliografia

Daniel Jurafsky & James H. Martin.

Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition**.** Pearson/Prentice Hall.



2000



2009

2019?

Stanford University

University of Colorado, Boulder

Bibliografia – Capítulo 5

Speech and Language Processing (3rd ed. draft)

Dan Jurafsky and James H. Martin

Appendix Chapters (likely just on the web)

A: Hidden Markov Models

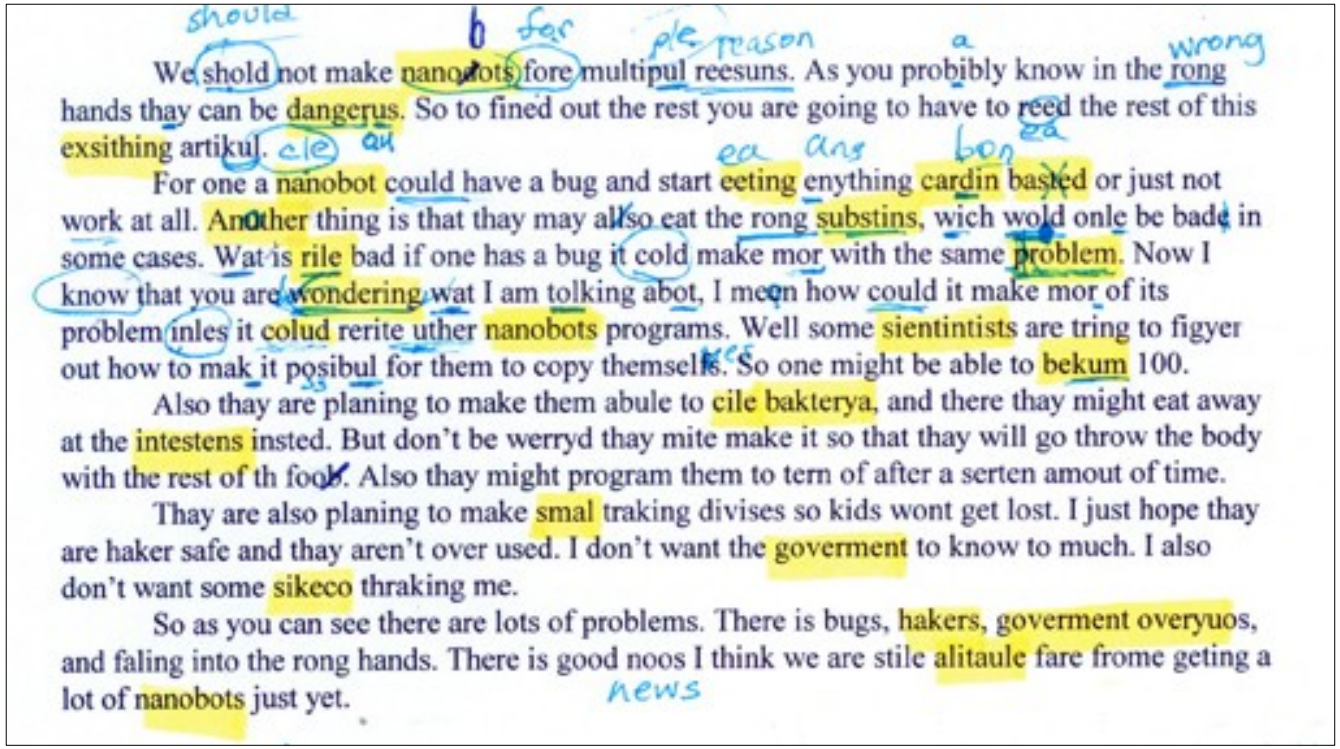
B: Spelling Correction and the Noisy Channel

C: Computing with Word Senses: WSD and WordNet

Intro, Sim [pptx] [pdf]
WSD [pptx] [pdf]

[expanded from parts of Ch. 19, 20 in 2nd ed]

https://web.stanford.edu/~jurafsky/slp3/



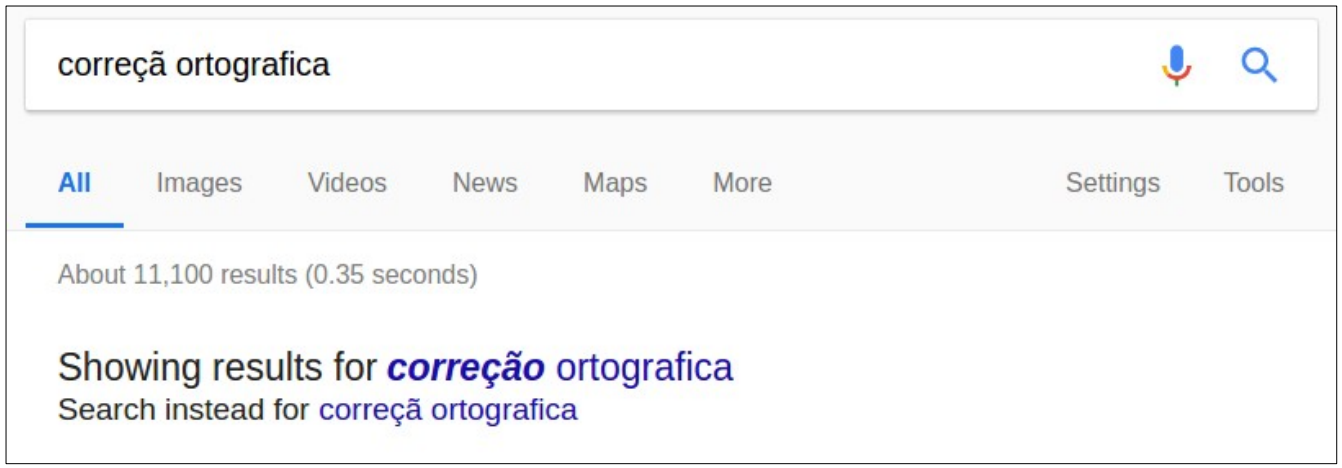
We shold not make nanobots fore multipul reesuns. As you probibly know in the rong hands thay can be dangerus. So to fined out the rest you are going to have to reed the rest of this exsithing artikul.

For one a nanobot could have a bug and start eeting enything cardin basted or just not work at all. Another thing is that thay may also eat the rong substins, wich wold onle be bade in some cases. Wat is rile bad if one has a bug it cold make mor with the same problem. Now I know that you are wondering wat I am tolking abot, I meen how could it make mor of its problem inles it colud rerite uther nanobots programs. Well some sientintists are tring to figyer out how to mak it posibul for them to copy themselfs. So one might be able to bekum 100.

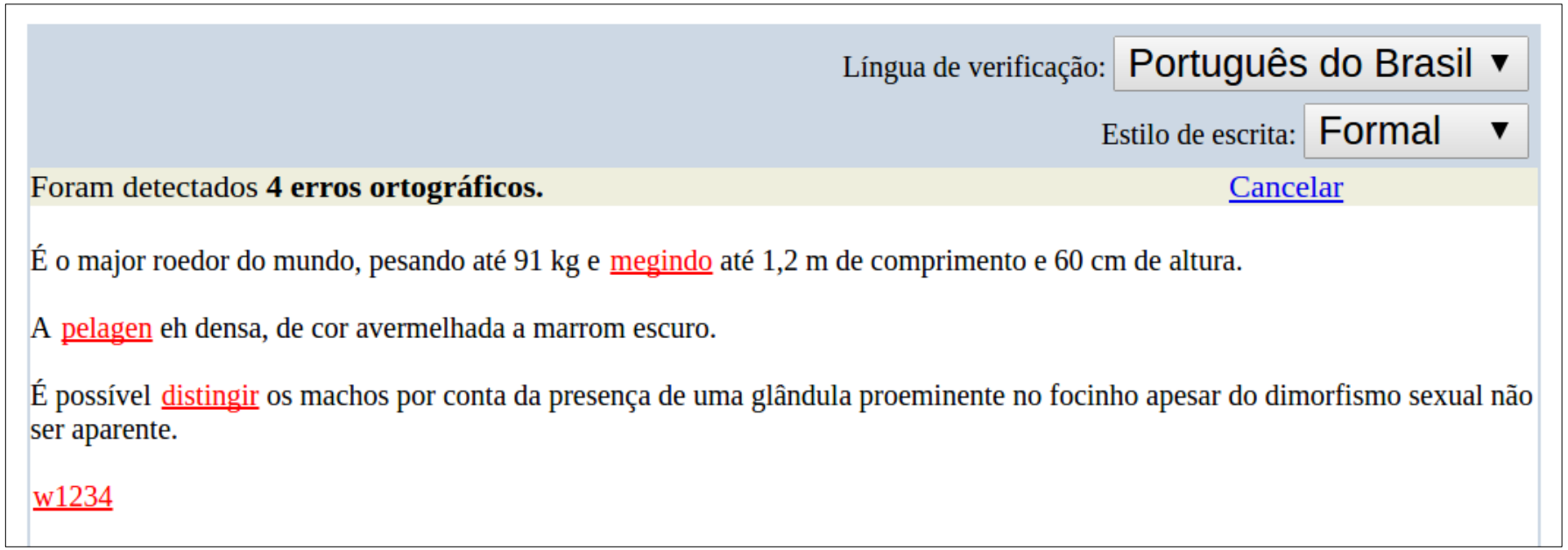
Also thay are planing to make them abule to cile bakterya, and there thay might eat away at the intestens insted. But don't be werryd thay mite make it so that thay will go throw the body with the rest of th foods. Also thay might program them to tern of after a serten amout of time.

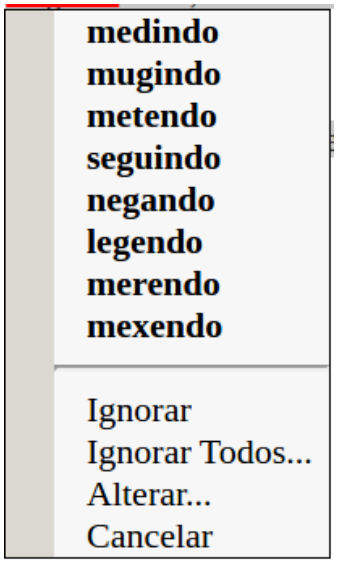
Thay are also planing to make smal traking divises so kids wont get lost. I just hope thay are haker safe and thay aren't over used. I don't want the goverment to know to much. I also don't want some sikeco thraking me.

So as you can see there are lots of problems. There is bugs, hakers, goverment overyuos, and faling into the rong hands. There is good noos I think we are stile alitaule fare frome geting a lot of nanobots just yet.



https://www.flip.pt/FLiP-On-line/Corrector-ortografico-e-sintactico





https://languagetool.org/pt-BR/

É o major roedor do mundo, pesando até 91 kg e megindo até 1,2 m de comprimento e 60 cm de altura.

A pelagen eh densa, de cor avermelhada a marrom escuro.

É possível distingir os machos por conta da presença de uma glândula proeminente no focinho apesar do dimorfismo sexual não ser aparente.

w1234

Possível erro ortográfico encontrado
medindo
mugindo
Megido
megafundo
Meguido
(outra substituição)
Ignorar erros para esta palavra
Adicionar ao dicionário…
Exemplos…

Nomenclatura

- ***Spell checking***: Correção ortográfica
- ***Spell checkers***: é o artefato (a ferramenta)

Nessa aula veremos algoritmos básicos para correção ortográfica

Tipos (classes) de erros

- **1) Erros associados a elementos que não são palavras de um vocabulário:**

  “Unversidade” → “Universidade”

  Erros fáceis de identificar

- **2) Erros associados a elementos que são palavras:**
  - Erros de tipografia (gralha):
    “oba” → “boa”
  - Erros cognitivos (palavras homófonas)
    “Conselho” → “Concelho”
    “One” → “Won”

Erros difíceis de identificar (Noisy channel model)

International Conference on Application of Natural Language to Information Systems
NLDB 2002: Natural Language Processing and Information Systems pp 183-190 | Cite as

Evaluating a Spelling Support in a Search Engine

Authors Authors and affiliations

Hercules Dalianis

Abstract

The information in a database is usually accessed using SQL or some other query language, but if one uses a free text retrieval system the retrieval of text based information becomes much easier and user friendly, since one can use natural languages techniques such as automatic spell checking and stemming. The free text retrieval system needs first to index the database but then it is just to search the database. Normally a search engine does not give any answers to queries when the search words does not exist in the index, therefore we connected a spell checker module into a search engine and evaluated it. The domain used was the web site of the Swedish National Tax Board (Riksskatteverket, RSV), where the search engine was used between April and Sept 2001. One million queries were made by the public. Of these queries 10 percent were “misspelled” or erroneous and our spell checker corrected around 90 percent of these.

Tarefas de ortografia

- **Detecção** de erros de ortografia.
  - **Erro tipo 1:** Usando um “bom” dicionário.
  - **Erro tipo 2:** Usando o *contexto*.

- **Correção** de erros de ortografia:
  - **Erro tipo 1:** Gerar palavras candidatas (que sejam palavras válidas) e selecionar a “melhor” candidata.
  - **Erro tipo 2:** Gerar palavras candidatas (com pronunciação ou ortografia similar) e selecionar o “melhor” candidata.

Modelo de canal ruidoso para correção ortográfica

The noisy channel model of spelling

Modelo de canal ruidoso

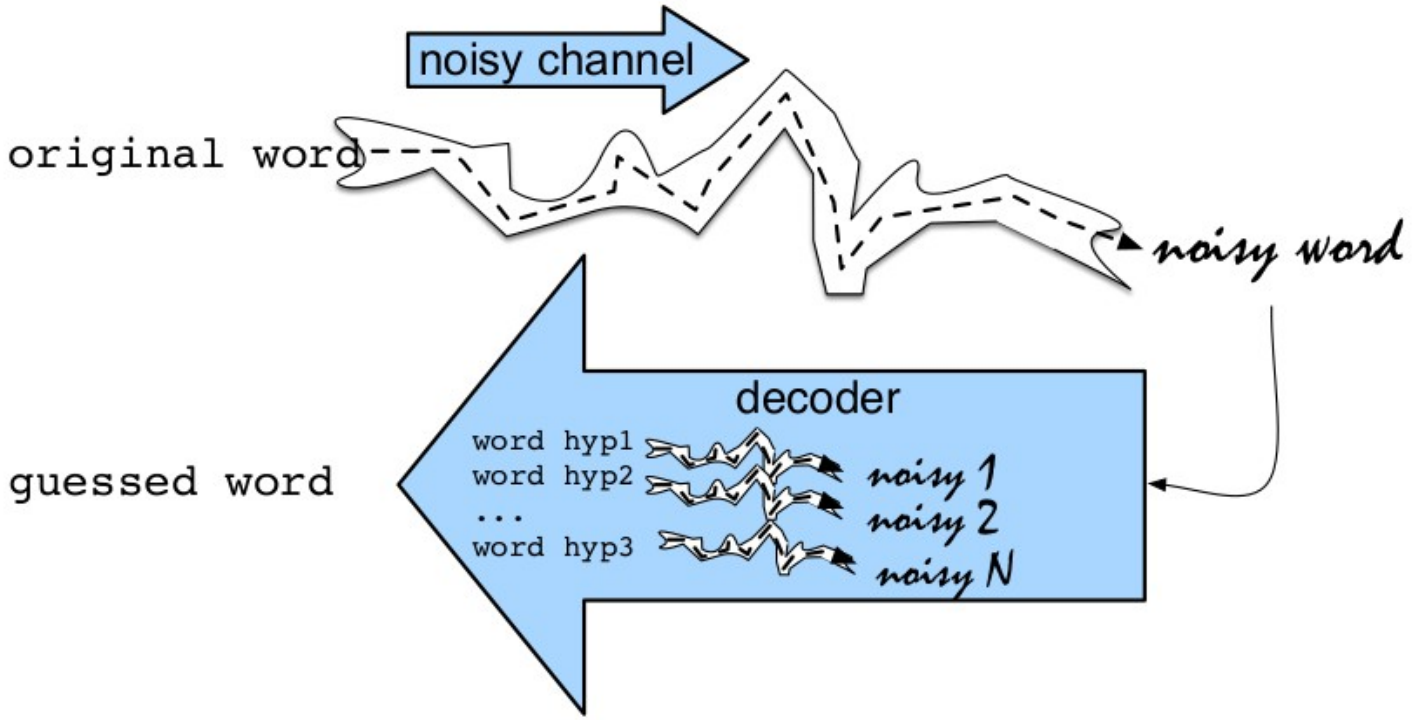
- Modelo muito utilizado em Processamento de Sinais.

- Assume-se que existe um **sinal original** que, ao **passar** por um canal (ruidoso) se **corrompe**.

  Tendo no final um sinal com ruído.

  Assim, é necessário recuperar (descodificar) o sinal original.

Modelo de canal ruidoso



Como modelar esse canal?

Modelo de canal ruidoso

- É um tipo de **inferência de Bayes**.

- Ao observar uma palavra ***x*** (i.e., uma palavra com erro ortográfico) queremos encontrar uma palavra ***w*** (de um vocabulário ***V***) que gerou a palavra com erro.

$$\hat{w} = \underset{w \in V}{\operatorname{argmax}} P(w|x)$$

Palavra com erro ortográfico

$\hat{w}$ é a palavra que maximiza $P(w|x)$

Modelo de canal ruidoso

- Segundo a regra de Bayes:

$$P(a|b) = \frac{P(b|a)P(a)}{P(b)}$$

$$\hat{w} = \underset{w \in V}{\operatorname{argmax}} P(w|x)$$

Modelo de canal ruidoso

- Segundo a regra de Bayes:

$$P(a|b) = \frac{P(b|a)P(a)}{P(b)}$$

$$\hat{w} = \underset{w \in V}{\operatorname{argmax}} P(w|x) = \underset{w \in V}{\operatorname{argmax}} \frac{P(x|w)P(w)}{P(x)}$$

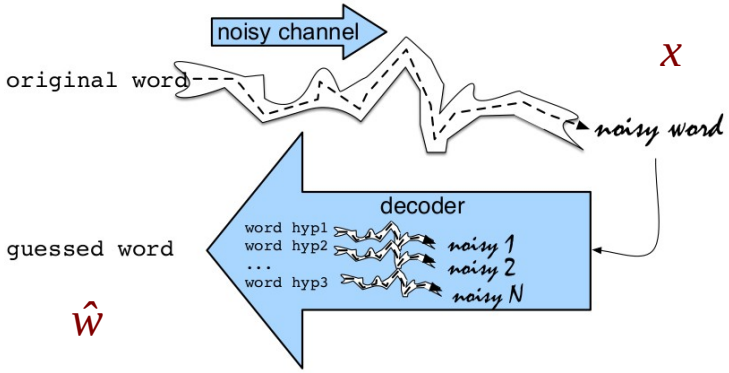
Palavra com erro ortográfico

Palavra com erro ortográfico

Modelo de canal ruidoso

- Segundo a regra de Bayes:

$$P(a|b) = \frac{P(b|a)P(a)}{P(b)}$$



$$\hat{w} = \underset{w \in V}{\operatorname{argmax}} P(w|x) = \underset{w \in V}{\operatorname{argmax}} \frac{P(x|w)P(w)}{P(x)}$$

$$= \underset{w \in V}{\operatorname{argmax}} P(x|w)\,P(w)$$

Modelo de canal ruidoso

x: Palavra com erro ortográfico

$$\hat{w} = \underset{w \in C}{\operatorname{argmax}} \overbrace{P(x|w)}^{\text{channel model}} \overbrace{P(w)}^{\text{prior}}$$

Probabilidade à priori
(modelo de linguagem – unigrama, bigrama, trigrama, ...)

Probabilidade de x ter sido gerado por w
(modelo do canal | modelo do erro)

Modelo de canal ruidoso para correção ortográfica proposta nos 1990

- **IBM**

  Mays, Eric, Fred J. Damerau and Robert L. Mercer. 1991. Context based spelling correction. Information Processing and Management, 23(5), 517–522.

- **AT&T Bell Labs**

  Kernighan, Mark D., Kenneth W. Church, and William A. Gale. 1990. A spelling correction program based on a noisy channel model. Proceedings of COLING 1990, 205-210.

Exemplo de uma sequência de caracteres que não é uma palavra de um vocabulário

acress

Exemplo de uma sequência de caracteres que não é uma palavra de um vocabulário

acress

Seleção de palavras candidatas

Podemos considerar

- Palavras com similar ortografia

  ie., palavras com uma distância de edição pequena

  actress, across, access, ...

- Palavras com similar pronunciação

  ie., palavras com uma distância de pronunciação pequena.

Exemplo de uma sequência de caracteres que não é uma palavra de um vocabulário

acress

Distância de edição (Demeraou-Levenshtein)

Considera a distância mínima entre 2 cadeias, em que a edição pode ser:

- **Inserção** de um caractere
- **Eliminação** de um caractere
- **Substituição** de um caractere
- **Transposição** de 2 caracteres

Exemplo de uma sequência de caracteres que não é uma palavra de um vocabulário

		Transformation			
Error	Correction	Correct Letter	Error Letter	Position (Letter #)	Type
acress	actress	t	—	2	deletion
acress	cress	—	a	0	insertion
acress	caress	ca	ac	0	transposition
acress	access	c	r	2	substitution
acress	across	o	e	3	substitution
acress	acres	—	s	5	insertion
acress	acres	—	s	4	insertion

x w Considerando w e aplicando uma operação de edição podemos ter x

80% dos erros estão a uma distância de 1.
Quase todos os erros estão a uma distância de 2.

Exemplo de uma sequência de caracteres que não é uma palavra de um vocabulário

$$\hat{w} = \underset{w \in C}{\operatorname{argmax}} \overbrace{P(x|w)}^{\text{channel model}} \overbrace{P(w)}^{\text{prior}}$$

Considerando um **modelo de linguagem** baseado em unigramas e 404 253 213 palavras extraídas do corpus de *Contemporary English* (COCA).

w	count(w)	p(w)
actress	9,321	.0000231
cress	220	.000000544
caress	686	.00000170
access	37,038	.0000916
across	120,844	.000299
acres	12,874	.0000318

Exemplo de uma sequência de caracteres que não é uma palavra de um vocabulário

$$\hat{w} = \underset{w \in C}{\operatorname{argmax}} \overbrace{P(x|w)}^{\text{channel model}} \overbrace{P(w)}^{\text{prior}}$$

Palavra com erro ortográfico

O modelo do canal pode ser calculado considerando o número de vezes que uma letra foi substituída por outra usando um corpus de erros.

Por exemplo o número de vezes que a letra ‘e’ for substituída pela letra ‘o’ em P(acress | across)

Podemos construir uma **matriz de confusão** para contar esses erros.

Exemplo de uma sequência de caracteres que não é uma palavra de um vocabulário

Kernighan, Mark D., Kenneth W. Church, and William A. Gale. 1990. A spelling correction program based on a noisy channel model. Proceedings of COLING 1990, 205-210.

Se $x = x_1, x_2, x_3, \ldots, x_n$

$w = w_1, w_2, w_3, \ldots, w_m$

P(x | w) = Probabilidade da edição de w para torna-la em x

Quatro matrizes de confusão

Número de vezes que y é apagado após o x

```
del[x, y]:   count(xy typed as x)
ins[x, y]:   count(x typed as xy)
sub[x, y]:   count(x typed as y)
trans[x, y]: count(xy typed as yx)
```

Número de vezes que y é inserido após o x

Exemplo de uma sequência de caracteres que não é uma palavra de um vocabulário

Erro: ‘e’ foi escrita como ‘a’

Erro: ‘a’ foi escrita como ‘e’

sub[X, Y] = Substitution of X (incorrect) for Y (correct)

X \ Y (correct)	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	0	7	1	342	0	0	2	118	0	1	0	0	3	76	0	0	1	35	9	9	0	1	0	5	0
b	0	0	9	9	2	2	3	1	0	0	0	5	11	5	0	10	0	0	2	1	0	0	8	0	0	0
c	6	5	0	16	0	9	5	0	0	0	1	0	7	9	1	10	2	5	39	40	1	3	7	1	1	0
d	1	10	13	0	12	0	5	5	0	0	2	3	7	3	0	1	0	43	30	22	0	0	4	0	2	0
e	388	0	3	11	0	2	2	0	89	0	0	3	0	5	93	0	0	14	12	6	15	0	1	0	18	0
f	0	15	0	3	1	0	5	2	0	0	0	3	4	1	0	0	0	6	4	12	0	0	2	0	0	0
g	4	1	11	11	9	2	0	0	0	1	1	3	0	0	2	1	3	5	13	21	0	0	1	0	3	0
h	1	8	0	3	0	0	0	0	0	0	2	0	12	14	2	3	0	3	1	11	0	0	2	0	0	0
i	1	0	0	0	146	0	1	0	0	0	0	6	0	0	49	0	0	0	2	1	47	0	2	1	15	0
j	0	0	0	0	0	0	1	0	0	0	0	2	1	0	0	0	0	0	5	0	0	0	0	0	0	0
k	1	2	8	4	1	1	2	5	0	0	0	0	5	0	2	0	0	0	6	0	0	0	4	0	0	3
l	2	10	1	4	0	4	5	6	13	0	1	0	0	14	2	5	0	11	10	2	0	0	0	0	0	0
m	1	3	7	8	0	2	0	6	0	0	4	4	0	180	0	6	0	0	9	15	13	3	2	2	3	0
n	2	7	6	5	3	0	1	19	1	0	4	35	78	0	0	7	0	28	5	7	0	0	1	2	0	2
o	91	1	1	3	116	0	0	0	25	0	2	0	0	0	0	14	0	2	4	14	39	0	0	0	18	0
p	0	11	1	2	0	6	5	0	2	9	0	2	7	6	15	0	0	1	3	6	0	4	1	0	0	0
q	0	0	1	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	0	14	0	30	12	2	2	8	2	0	5	8	4	20	1	14	0	0	12	22	4	0	0	1	0	0
s	11	8	27	33	35	4	0	1	0	1	0	27	0	6	1	7	0	14	0	15	0	0	5	3	20	1
t	3	4	9	42	7	5	19	5	0	1	0	14	9	5	5	6	0	11	37	0	0	2	19	0	7	6
u	20	0	0	0	44	0	0	0	64	0	0	0	0	2	43	0	0	4	0	0	0	0	2	0	8	0
v	0	0	7	0	0	3	0	0	0	0	0	1	0	0	1	0	0	0	8	3	0	0	0	0	0	0
w	2	2	1	0	1	0	0	2	0	0	1	0	0	0	0	7	0	6	3	3	1	0	0	0	0	0
x	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	0	0	0	0	0	0
y	0	0	2	0	15	0	1	7	15	0	0	0	2	0	6	1	0	7	36	8	5	0	0	1	0	0
z	0	0	0	7	0	0	0	0	0	0	0	7	5	0	0	0	0	2	21	3	0	0	0	0	3	0

Matriz de confusão

http://norvig.com/ngrams/spell-errors.txt

```
raining: rainning, raning
writings: writtings
disparagingly: disparingly
yellow: yello
four: forer, fours, fuore, fore*5, for*4
woods: woodes
hanging: haing
aggression: agression
looking: loking, begining, luing, look*2, locking, lucking, louk, looing, lookin, liking
eligible: eligble, elegable, eligable
electricity: electrisity, electricty*2, electrizity
scold: schold, skold
adaptable: adabtable
caned: canned, cained
immature: imature
shouldn't: shoudln, shouldnt
swivel: swival
appropriation: apropriation
fur: furr, fer
stabbed: stabed
Southwold: Suothwode
disturb: distrebe, desturb
recollections: reclections, recolections
prize: prise, prizer
wednesday: wensday, wedensday
succession: sucession, sucesion, succesion
straight: strate, strait, staidght, stright*2
guardsmen: gards_men
incremented: increented
bacon: backen, baken
pulse: pluse
elegant: elagent, elligit
second: secand, sexeon, secund, seconnd, seond, sekon
jamjar: jam_jar
sailed: saled, saild
blouse: boludes
thunder: thounder
cooking: coking, chocking, kooking, cocking
```

Matriz de confusão

https://en.wikipedia.org/wiki/Wikipedia:Lists_of_common_misspellings/For_machines

```
abandonned->abandoned
aberation->aberration
abilityes->abilities
abilties->abilities
abilty->ability
abondon->abandon
abbout->about
abotu->about
abouta->about a
aboutit->about it
aboutthe->about the
abscence->absence
abondoned->abandoned
abondoning->abandoning
abondons->abandons
aborigene->aborigine
accesories->accessories
accidant->accident
abortificant->abortifacient
abreviate->abbreviate
abreviated->abbreviated
abreviation->abbreviation
abritrary->arbitrary
absail->abseil
absailing->abseiling
absense->absence
absolutly->absolutely
absorbsion->absorption
absorbtion->absorption
abudance->abundance
abundacies->abundances
abundancies->abundances
```

Exemplo de uma sequência de caracteres que não é uma palavra de um vocabulário

$$\hat{w} = \underset{w \in C}{\operatorname{argmax}} \overbrace{P(x|w)}^{\text{channel model}} \overbrace{P(w)}^{\text{prior}}$$

Palavra com erro ortográfico

$$P(x|w) = \begin{cases} \dfrac{\text{del}[x_{i-1}, w_i]}{\text{count}[x_{i-1} w_i]}, & \text{if deletion} \\ \dfrac{\text{ins}[x_{i-1}, w_i]}{\text{count}[w_{i-1}]}, & \text{if insertion} \\ \dfrac{\text{sub}[x_i, w_i]}{\text{count}[w_i]}, & \text{if substitution} \\ \dfrac{\text{trans}[w_i, w_{i+1}]}{\text{count}[w_i w_{i+1}]}, & \text{if transposition} \end{cases}$$

Exemplo de uma sequência de caracteres que não é uma palavra de um vocabulário

Kernighan, Mark D., Kenneth W. Church, and William A. Gale. 1990. A spelling correction program based on a noisy channelmodel. Proceedings of COLING 1990, 205-210.

Candidate Correction	Correct Letter	Error Letter	x|w	P(x|w)	P(w)	10^9*P(x|w)P(w)
actress	t	-	c|ct	.000117	.0000231	2.7
cress	-	a	a|#	.00000144	.000000544	0.00078
caress	ca	ac	ac|ca	.00000164	.00000170	0.0028
access	c	r	r|c	.000000209	.0000916	0.019
across	o	e	e|o	.0000093	.000299	2.8
acres	-	s	es|e	.0000321	.0000318	1.0
acres	-	s	ss|s	.0000342	.0000318	1.0

Exemplo de uma sequência de caracteres que não é uma palavra de um vocabulário

. . . “stellar and versatile **acress** whose combination of sass and glamour has defined her. . . ”

Candidate Correction	Correct Letter	Error Letter	x|w	P(x|w)	P(w)	10^9*P(x|w)P(w)
actress	t	-	c|ct	.000117	.0000231	2.7
cress	-	a	a|#	.00000144	.000000544	0.00078
caress	ca	ac	ac|ca	.00000164	.00000170	0.0028
access	c	r	r|c	.000000209	.0000916	0.019
across	o	e	e|o	.0000093	.000299	2.8
acres	-	s	es|e	.0000321	.0000318	1.0
acres	-	s	ss|s	.0000342	.0000318	1.0

Exemplo de uma sequência de caracteres que não é uma palavra de um vocabulário

. . . “stellar and versatile **acress** whose combination of sass and glamour has defined her. . . ”

O recomendável é usar como **modelo de linguagem** mais do que unigramas, por exemplo, considerando bigramas:

$$
\begin{aligned}
P(\text{actress}|\text{versatile}) &= .000021 \\
P(\text{across}|\text{versatile}) &= .000021 \\
P(\text{whose}|\text{actress}) &= .0010 \\
P(\text{whose}|\text{across}) &= .000006
\end{aligned}
$$

$$
\begin{aligned}
P(\text{“versatile actress whose”}) &= .000021 * .0010 = 210 \times 10^{-10} \\
P(\text{“versatile across whose”}) &= .000021 * .000006 = 1 \times 10^{-10}
\end{aligned}
$$

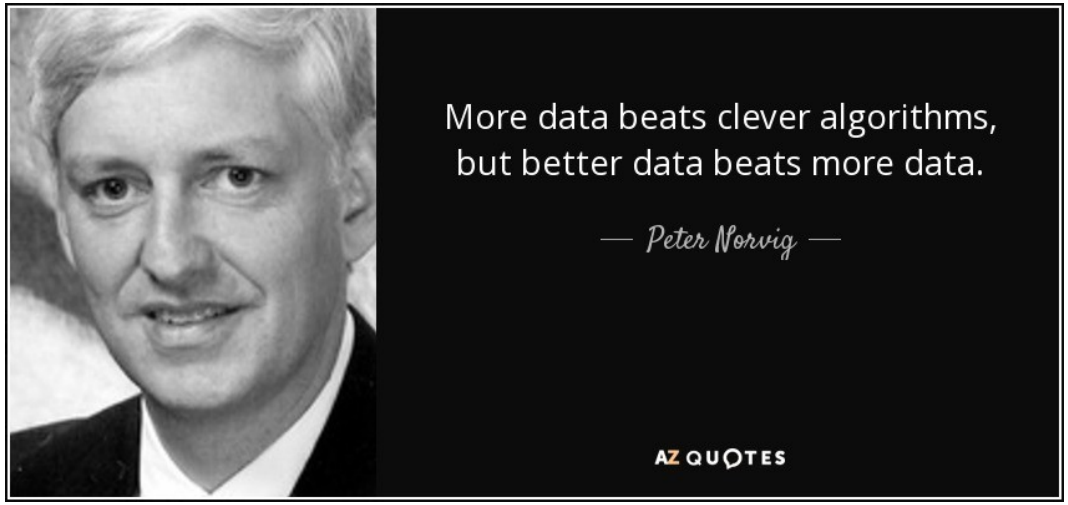
Correção ortográfica de elementos que são palavras

Correção ortográfica

Para cada palavra na frase:

- Gerar um conjunto de candidatos
- Selecionar os melhores candidatos

Peter Norvig



Born	December 14, 1956 (age 61)
Nationality	American
Alma mater	Brown University University of California, Berkeley
Known for	*Artificial Intelligence: A Modern Approach* *Paradigms of AI Programming: Case Studies in Common Lisp*
Website	www.norvig.com
	Scientific career
Fields	Computer Science
Institutions	Google Ames Research Center University of Southern California Brown University University of California, Berkeley
Thesis	*A Unified Theory of Inference for Text Understanding* (1986)
Doctoral advisor	Robert Wilensky[1]

Signature

Uma breve descrição dos layouts de teclado

Tipos (classes) de erros

- **1) Erros associados a elementos que não são palavras de um vocabulário:**

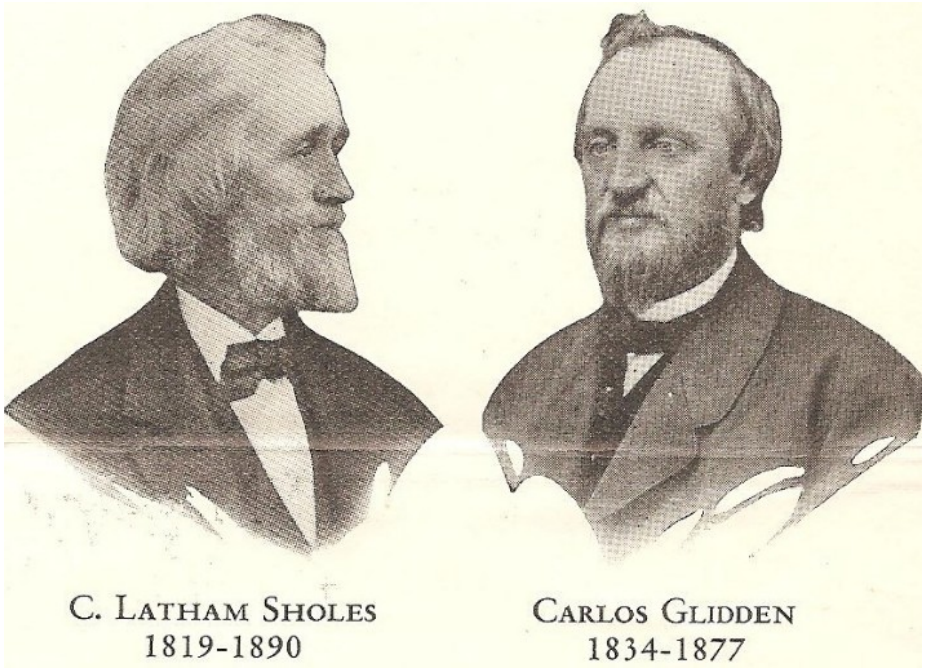
  “Unversidade” → “Universidade”

  Erros fáceis de identificar

- **2) Erros associados a elementos que são palavras:**
  - Erros de tipografia (gralha):
    “oba” → “boa”
  - Erros cognitivos (palavras homófonas)
    “Conselho” → “Concelho”
    “One” → “Won”

  Erros difíceis de identificar (Noisy channel model)

Primeira máquina que escrever de “sucesso”



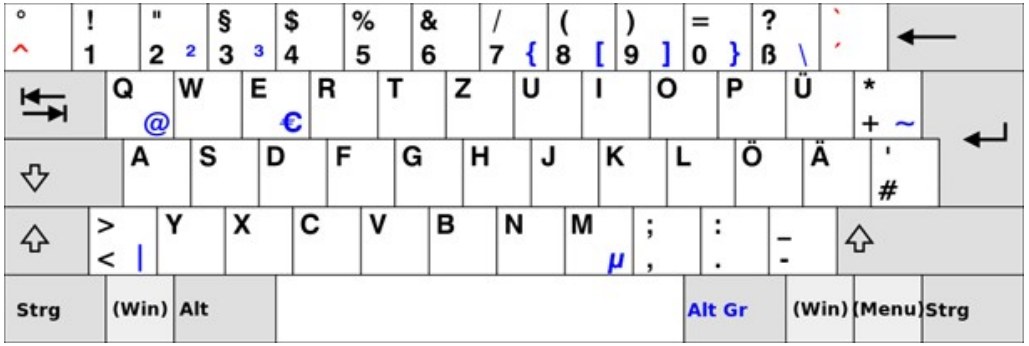
23456789-,
QWE.TYIUOP
ZSDFGHJKLM
AX&CVBN?;R

Sholes and Glidden typewriter - Remington (~1874)

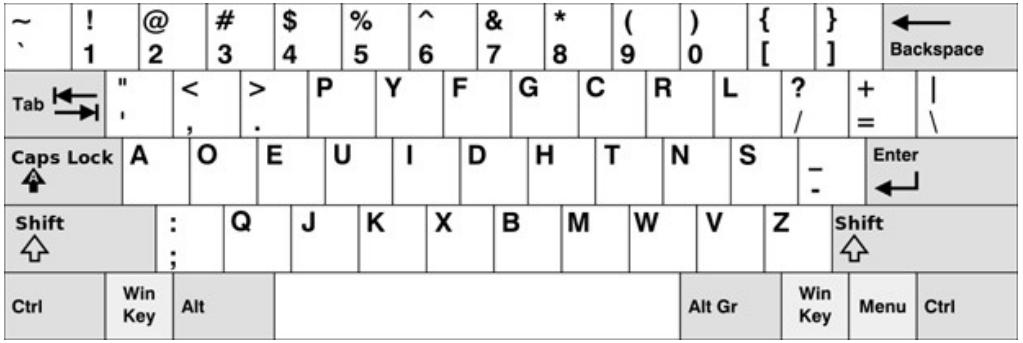
Layouts de teclado

Alguns layouts de teclado

Azerty



Qwertz



Dvorak (1936)

Considerações finais

1) Spell Checkers e seus braços

Utilizados não somente para tratamento de erros de digitação:

- Identificação de abreviações
- Vocabulário informal

Existem outros braços dos **spell checkers** que podem considerar:

- Interação com os usuários
- Utilização de dispositivos com teclado reduzido
- Informações de redes sociais

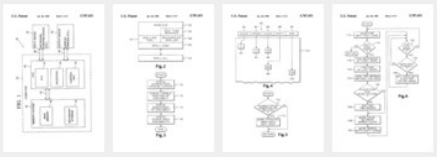
Method for background spell checking a word processing document

Method for background spell checking a word processing document

Abstract

A method for checking spelling in a word processor integrates spell checking with the editing process. During idle periods of the word processor, the spell checker scans an open document, and maintains a table of spelling status data, including codes to identify checked, unchecked, or edited ranges of characters. Spelling errors can be highlighted during an editing session. Spelling status data is maintained with the document so that spell checked portions of the document do not need to be re-checked.

Images (4)



Classifications

G06F17/273 Orthographic correction, e.g. spelling checkers, vowelisation

US5787451A
US Grant

Download PDF Find Prior Art

Inventor: Alex Mogilevsky
Current Assignee: Microsoft Technology Licensing LLC
Original Assignee: Microsoft Corp

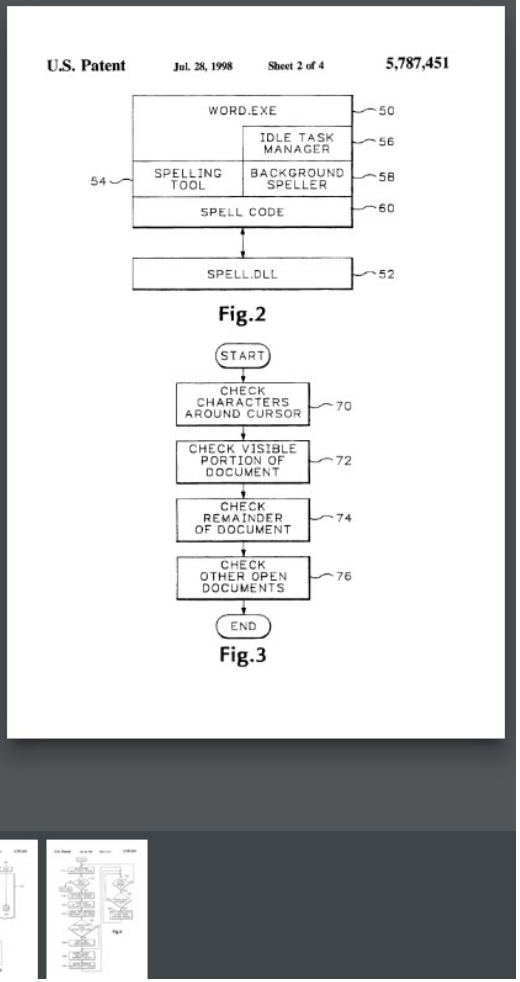
Priority date: 1995-05-08

Family: US (2)

Date	App/Pub Number	Status
1995	US08437949	Expired - Lifetime
1997-02-27	US08807624	Expired - Lifetime
1998-07-28	US5787451A	Grant

Info: Patent citations (6), Cited by (86), Legal events, Similar documents, Priority and Related Applications

External links: USPTO, USPTO Assignment, Espacenet, Global Dossier, Discuss



Description

This is a continuation of application Ser. No. 08/437,949, filed May 8, 1995, now U.S. Pat. No. 5,649,222.

The invention relates to word processing, and more specifically relates to a spell checking feature in a word processing system.

Claims (15)

I claim:

1. A computer-readable medium having computer-executable instructions for performing steps comprising:

(a) storing spelling status data to monitor spelling status of ranges of

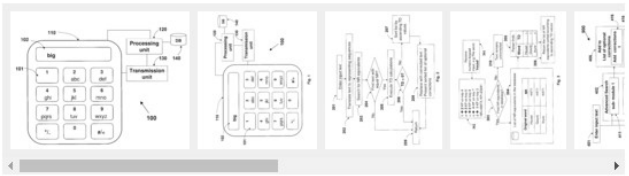
Spell checker for input of reduced keypad devices

Spell checker for input of reduced keypad devices

Abstract

The present invention discloses a system and a method for spell checking of an input text inserted by a user through a multiple-typing reduced numeric keypad device. The correction may be based on comparison to at least one given reference database vocabulary containing words and associated representation sequences, where the representation sequences may be: (i) Numeric Representations (NR) and Key Presses (KP) sequences. The NR is a sequence representing the digits numbers required to achieve a given word where the KP represents the sequence of the number of key presses required to reach each character in a key. For example sequences of the word "big" using a standard mobile phone reduced keypad: NR=244 and KP=231.

Images (10)



Classifications

G06F17/273 Orthographic correction, e.g. spelling checkers, vowelisation

US20080133222A1
US Application

Download PDF | Find Prior Art

Inventor: Yehuda Kogan, Ofer Digly, Moshe Sivan, Shay Harari

Current Assignee: KOOKOO INTERNET TECHNOLOGIES Ltd

Original Assignee: KOOKOO INTERNET TECHNOLOGIES Ltd

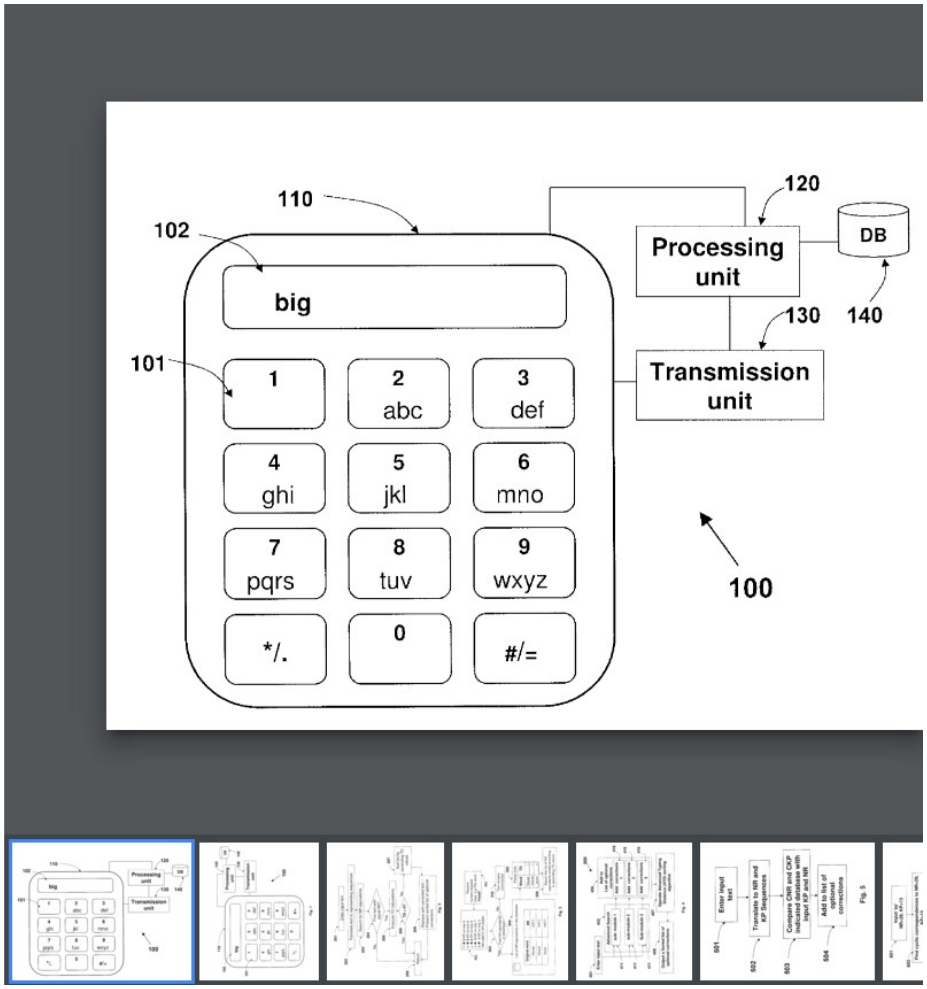
Priority date: 2006-11-30

Family: US (1)

Date	App/Pub Number	Status
2007-11-28	US11946443	Abandoned
2008-06-05	US20080133222A1	Application

Info: Patent citations (11), Cited by (14), Legal events, Similar documents, Priority and Related Applications

External links: USPTO, USPTO Assignment, Espacenet, Global Dossier, Discuss



Description

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit under 35 U.S.C. §119(e) of U.S. Provisional Patent Application No. 60/861,715 filed on Nov. 30, 2006, the content of which is incorporated by reference herein.

FIELD OF THE INVENTION

[0002] The present invention relates generally to the field of software tools

Claims (15)

1. A method for spell checking of an input text inserted by a user through a multiple-typing reduced numeric keypad of a communication device, wherein the correction is based on comparison to at least one given reference database vocabulary said method comprising the steps of:

translating the input text to representation sequences Numeric

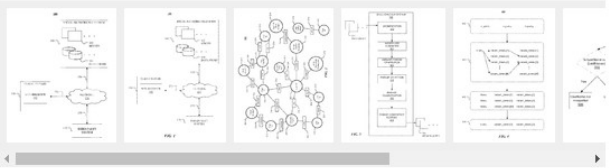
Social-Based Spelling Correction for Online Social Networks

Social-Based Spelling Correction for Online Social Networks

Abstract

In one embodiment, a method includes, receiving, from a client system of a user, a search query including n-grams. The method includes associating each n-gram with verticals based on an analysis of the n-grams by language models. The method includes determining, for each n-gram, if a bloom filter for a vertical associated with the n-gram indicates, based on sub-bloom filters of the bloom filter, the n-gram does exist or does not exist in a set of object names associated with the vertical. Each sub-bloom filter is associated with a subset of the set of object names and indicates the n-gram does exist or does not exist in its subset of object names. The method includes sending, to the client system, an indication that an n-gram of the n-grams is misspelled if a bloom filter indicates the n-gram does not exist in the set of object names associated with the vertical.

Images (8)



Classifications

G06F17/30867 Retrieval from the Internet, e.g. browsers by querying, e.g. search engines or meta-search engines, crawling techniques, push systems with filtering and personalisation

View 6 more classifications

US20170235842A1

US Application

Download PDF | Find Prior Art

Inventor: Ian Douglas Hegerty, Daniel Bernhardt, Feng Liang, Agnieszka Anna Podsiadlo

Current Assignee: Facebook Inc

Original Assignee: Facebook Inc

Priority date: 2014-12-01

Family: US (2)

Date	App/Pub Number	Status
2014	US14556368	Active
2017-05-01	US15583741	Pending
2017-08-17	US20170235842A1	Application

Info: Patent citations (110), Cited by (5), Similar documents, Priority and Related Applications

External links: USPTO, USPTO Assignment, Espacenet, Global Dossier, Discuss

Description

PRIORITY

[0001] This application is a continuation under 35 U.S.C. §120 of U.S. patent application Ser. No. 14/556,368, filed 1 Dec. 2014.

TECHNICAL FIELD

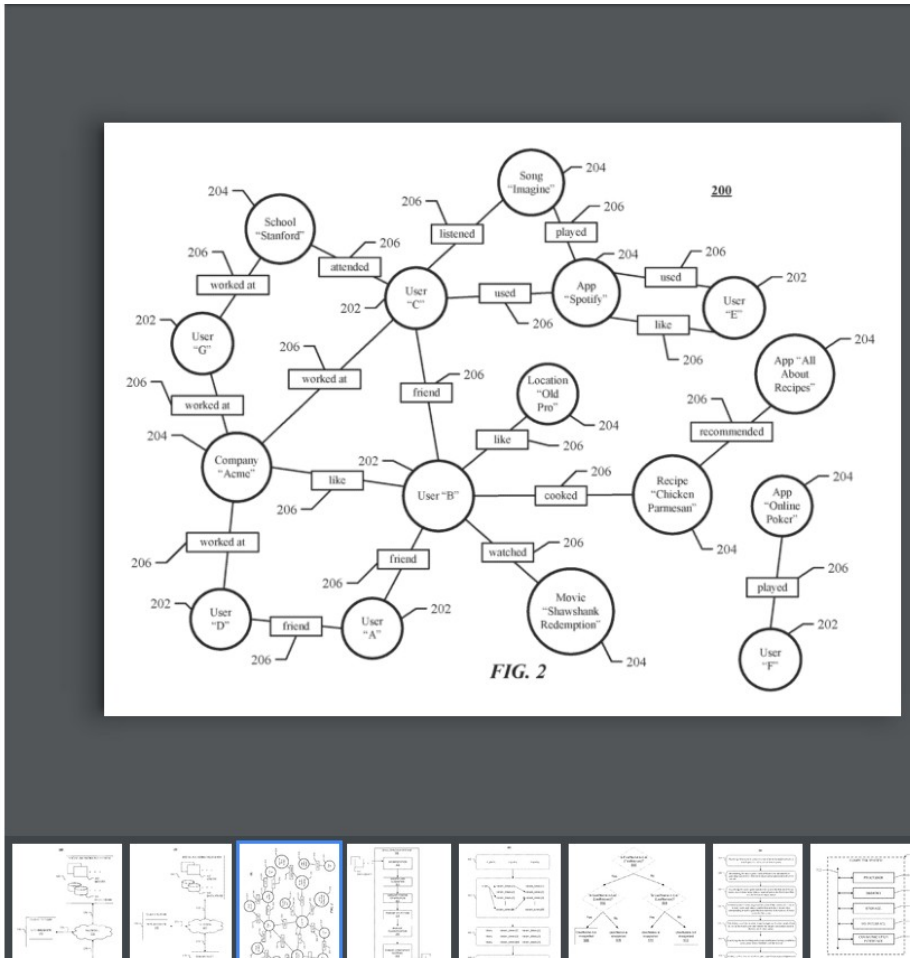
[0002] This disclosure generally relates to detecting and correcting

Claims (19)

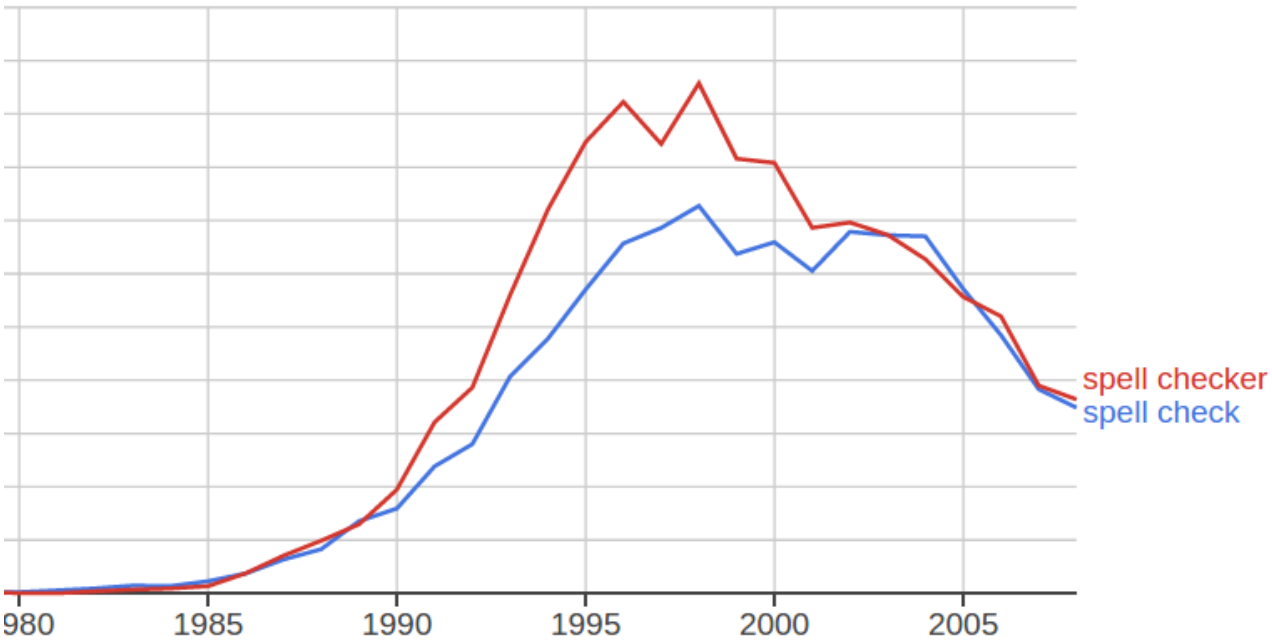
What is claimed is:

1. A method comprising, by one or more computing systems:

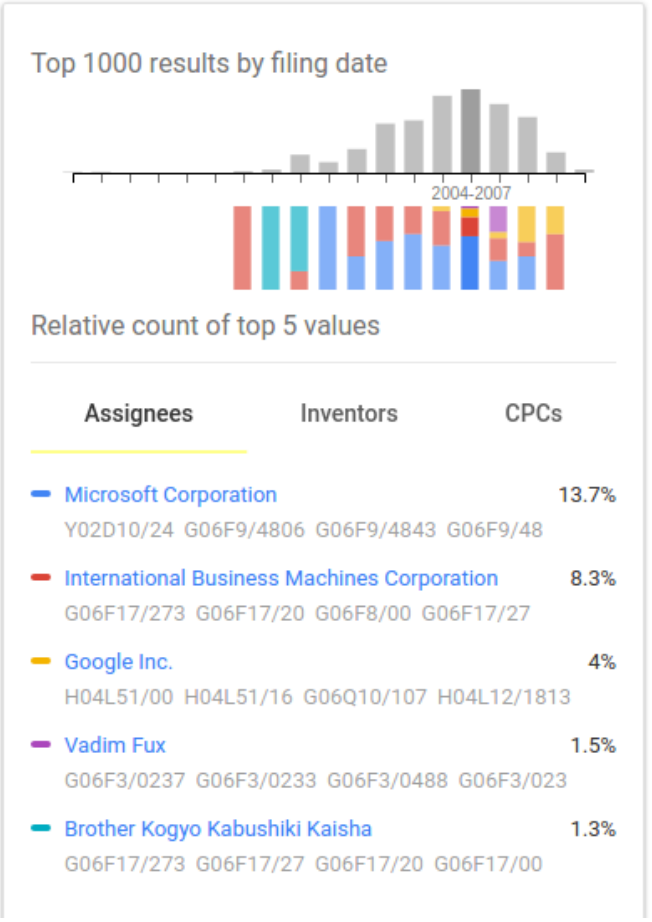
receiving, from a client system of a user, a search query comprising one or more n-grams:



2) Spell Checkers e estudos/aplicações



Google Ngram



Patentes