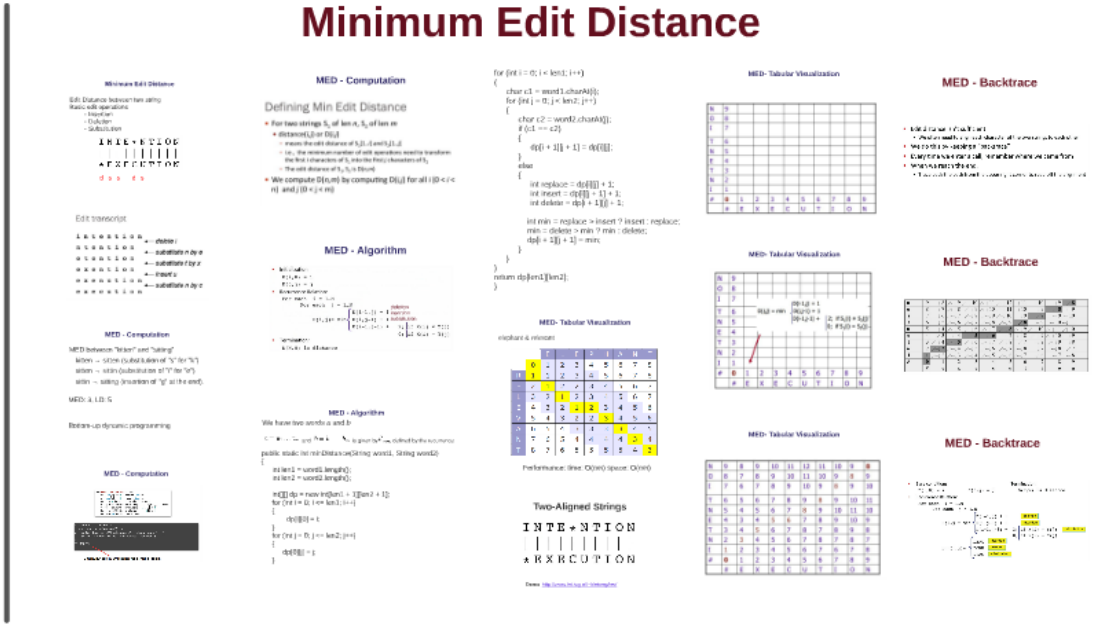
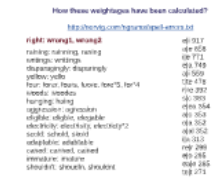
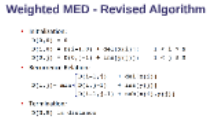
Spell Correction - Minimum Edit Distance

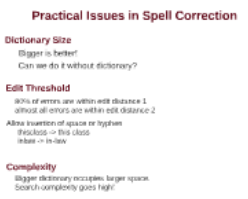


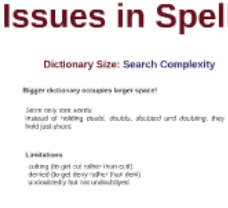
Weighted Edit Distance

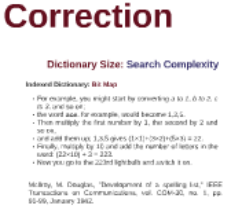


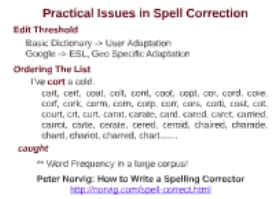


Practical Issues in Spell Correction

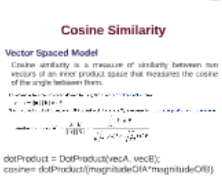


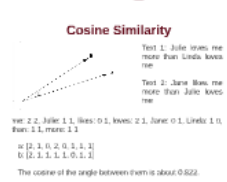


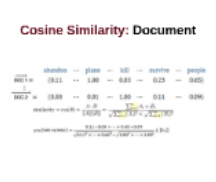


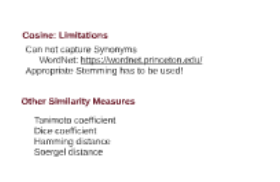


String Matching Paraphrase

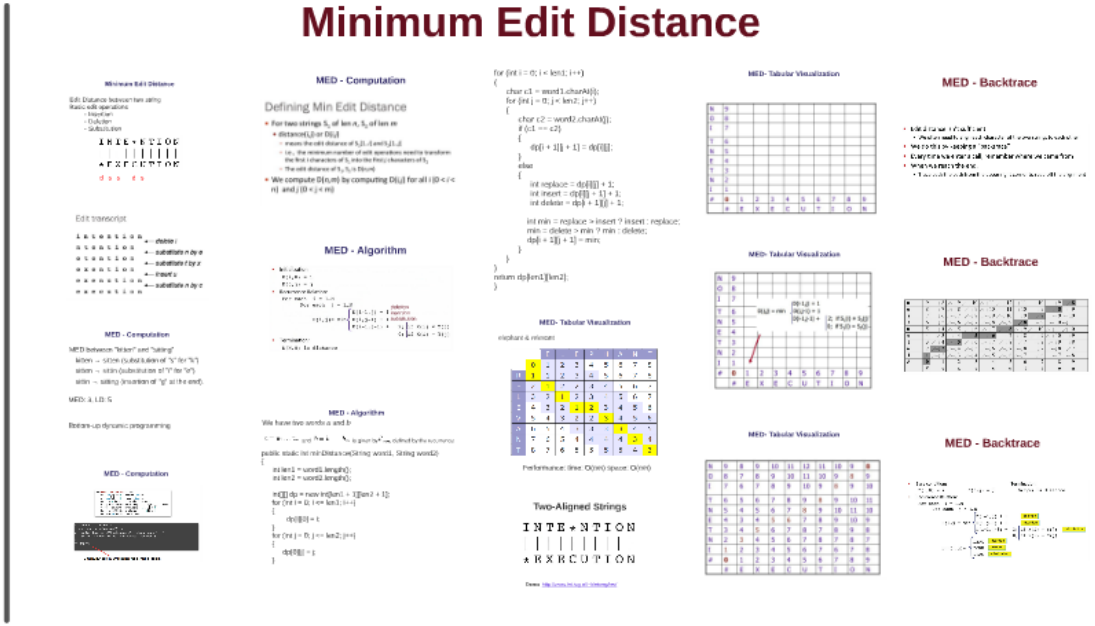




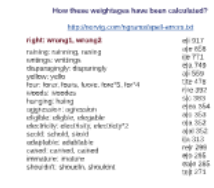


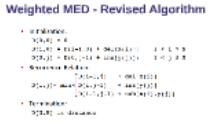


Spell Correction - Minimum Edit Distance

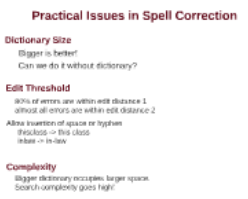


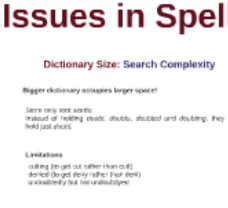
Weighted Edit Distance

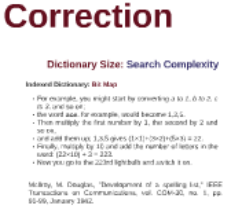


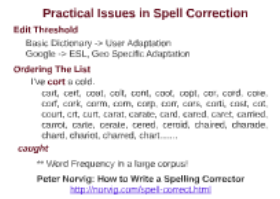


Practical Issues in Spell Correction

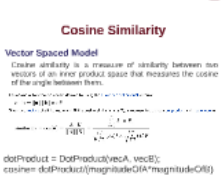


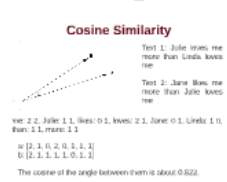


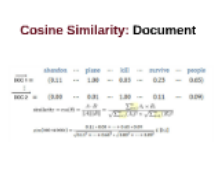


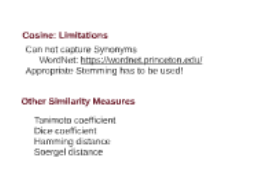


String Matching Paraphrase

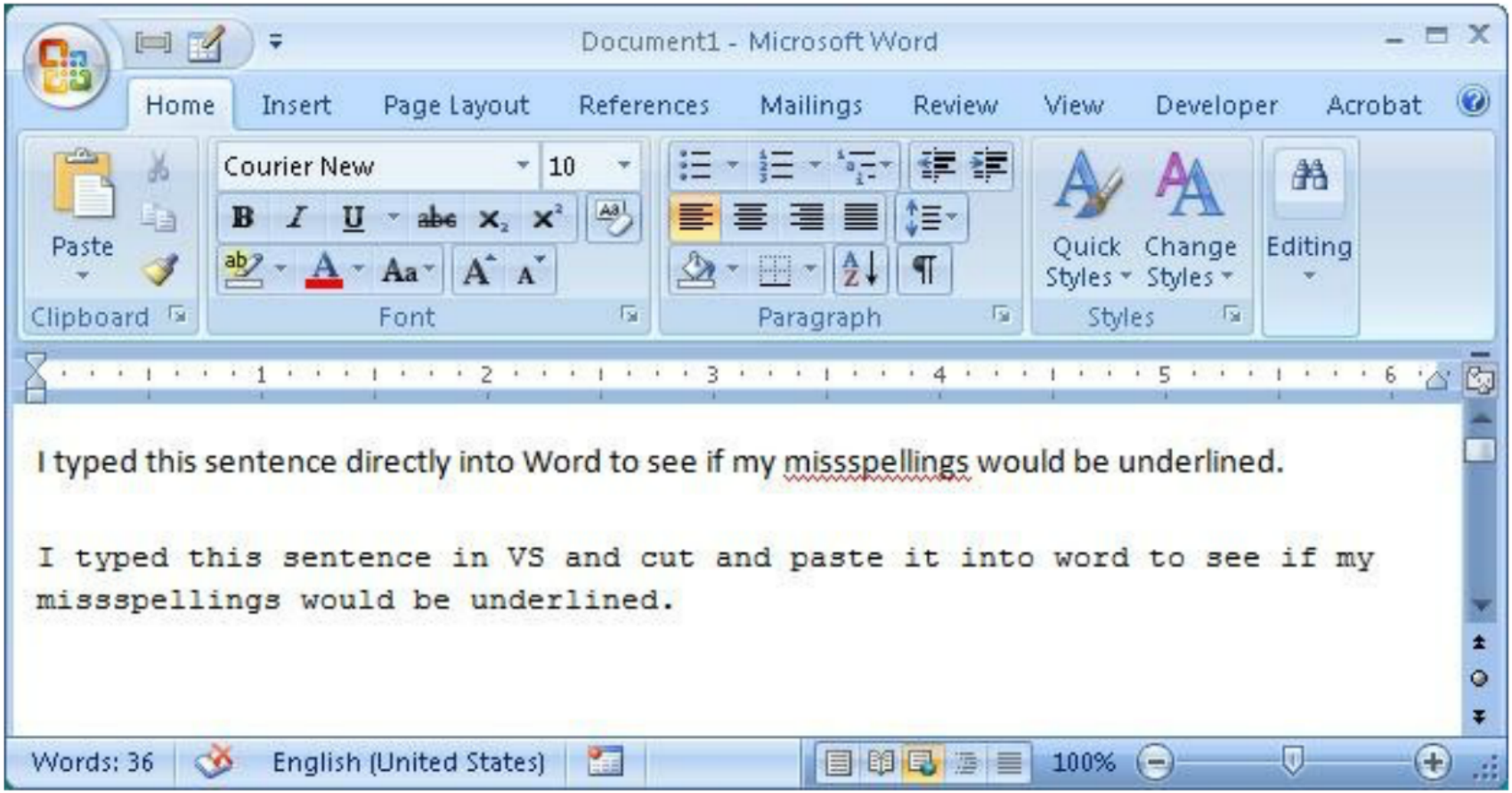








Spelling and Grammar Correction



Spelling Error - Types

Non-Word Error: ona -> on → **Out-Of-Dict Words**

Real-Word Error: there - > their

Cognitive Error: off - of

Language knowledge -> Language Models

26% Web Queries
40% in Social Media

Spelling Error Types

Non-Word Error: *ona {on,one,}*

Contextual Error: Win or loose, it was a great game!
{lose}

Grammar Error Types

Context-dependent Errors

I am [prepared → preparing] for the exam.

Lexical Choice Errors

I had a cup of [strong → powerful] tea

Function Words and Conjugation Errors

Can we discuss [null → about] this?

Let's go see [a → null] movie.

Preposition, Conjunction, Verb, Article,.....

ESL: English as Second Language

Spelling <Non-Word> Error Correction

User typed: *ona*

on
one
own
won
done
gone
stone
alone
phone
.....

Error Detection
Error Correction
- Replace
- Choose on from suggestions

Spelling Correction

A large dictionary is required

- But larger dict size means increase of search complexity!

Correction - Suggestions

Minimum Edit Distance (MED)
/ Levenshtein Distance
** Introduced by Vladimir Levenshtein in 1966

pell Correction - Minimum Edit Distance

Minimum Edit Distance

Minimum Edit Distance

Edit Distance between two string
Basic edit operations
- Insertion
- Deletion
- Substitution

```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
  d s s   d s
```

Edit transcript

```
i n t e n t i o n      ← delete i
n t e n t i o n        ← substitute n by e
e t e n t i o n        ← substitute t by x
e x e n t i o n        ← insert u
e x e n u t i o n      ← substitute n by c
e x e c u t i o n
```

MED - Computation

MED between "kitten" and "sitting"
kitten → sitten (substitution of "s" for "k")
sitten → sittin (substitution of "i" for "e")
sittin → sitting (insertion of "g" at the end).

MED: 3, LD: 5

Bottom-up dynamic programming

MED - Computation

MED - Computation

Defining Min Edit Distance

- For two strings S_1 of len n, S_2 of len m
  - distance(i,j) or D(i,j)
    - means the edit distance of $S_1[1..i]$ and $S_2[1..j]$
    - i.e., the minimum number of edit operations need to transform the first i characters of S_1 into the first j characters of S_2
    - The edit distance of S_1, S_2 is D(n,m)
- We compute D(n,m) by computing D(i,j) for all i ($0 < i < n$) and j ($0 < j < m$)

MED - Algorithm

- Initialization
  - D(i,0) = i
  - D(0,j) = j
- Recurrence Relation:
  For each i = 1..M
  For each j = 1..N
  $$D(i,j) = \min \begin{cases} D(i-1,j) + 1 & \text{deletion} \\ D(i,j-1) + 1 & \text{insertion} \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } X(i) \neq Y(j) \\ 0; & \text{if } X(i) = Y(j) \end{cases} & \text{substitution} \end{cases}$$
- Termination:
  D(N,M) is distance

MED - Algorithm

We have two words a and b

$a = a_1 \ldots a_n$ and $b = b_1 \ldots b_m$ is given by d_{mn}, defined by the recurrence

```
public static int minDistance(String word1, String word2)
{
    int len1 = word1.length();
    int len2 = word2.length();

    int[][] dp = new int[len1 + 1][len2 + 1];
    for (int i = 0; i <= len1; i++)
    {
        dp[i][0] = i;
    }
    for (int j = 0; j <= len2; j++)
    {
        dp[0][j] = j;
    }
```

```
for (int i = 0; i < len1; i++)
{
    char c1 = word1.charAt(i);
    for (int j = 0; j < len2; j++)
    {
        char c2 = word2.charAt(j);
        if (c1 == c2)
        {
            dp[i + 1][j + 1] = dp[i][j];
        }
        else
        {
            int replace = dp[i][j] + 1;
            int insert = dp[i][j + 1] + 1;
            int delete = dp[i + 1][j] + 1;

            int min = replace > insert ? insert : replace;
            min = delete > min ? min : delete;
            dp[i + 1][j + 1] = min;
        }
    }
}
return dp[len1][len2];
}
```

MED- Tabular Visualization

elephant & relevant

		E	L	E	P	H	A	N	T
	0	1	2	3	4	5	6	7	8
R	1	1	2	3	4	5	6	7	8
E	2	1	2	2	3	4	5	6	7
L	3	2	1	2	3	4	5	6	7
E	4	3	2	1	2	3	4	5	6
V	5	4	3	2	2	3	4	5	6
A	6	5	4	3	3	3	3	4	5
N	7	6	5	4	4	4	4	3	4
T	8	7	6	5	5	5	5	4	3

Performance: time: O(nm) space: O(nm)

Two-Aligned Strings

```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
```

Demo http://www.let.rug.nl/~kleiweg/lev/

MED- Tabular Visualization

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1									
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

MED- Tabular Visualization

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases}$$

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1									
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

MED- Tabular Visualization

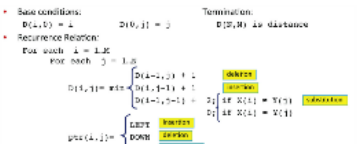
N	9	8	9	10	11	12	11	10	9	8
O	8	7	8	9	10	11	10	9	8	9
I	7	6	7	8	9	10	9	8	9	10
T	6	5	6	7	8	9	8	9	10	11
N	5	4	5	6	7	8	9	10	11	10
E	4	3	4	5	6	7	8	9	10	9
T	3	4	5	6	7	8	7	8	9	8
N	2	3	4	5	6	7	8	7	8	7
I	1	2	3	4	5	6	7	6	7	8
#	0	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

MED - Backtrace

- Edit distance isn't sufficient
  - We often need to align each character of the two strings to each other
- We do this by keeping a "backtrace"
- Every time we enter a cell, remember where we came from
- When we reach the end,
  - Trace back the path from the upper right corner to read off the alignment

MED - Backtrace

MED - Backtrace



Weighted Edit Distance

```
int w_med(String str1, String str2, int insert, int delete, int
substitute)
```

Minimum Edit Distance

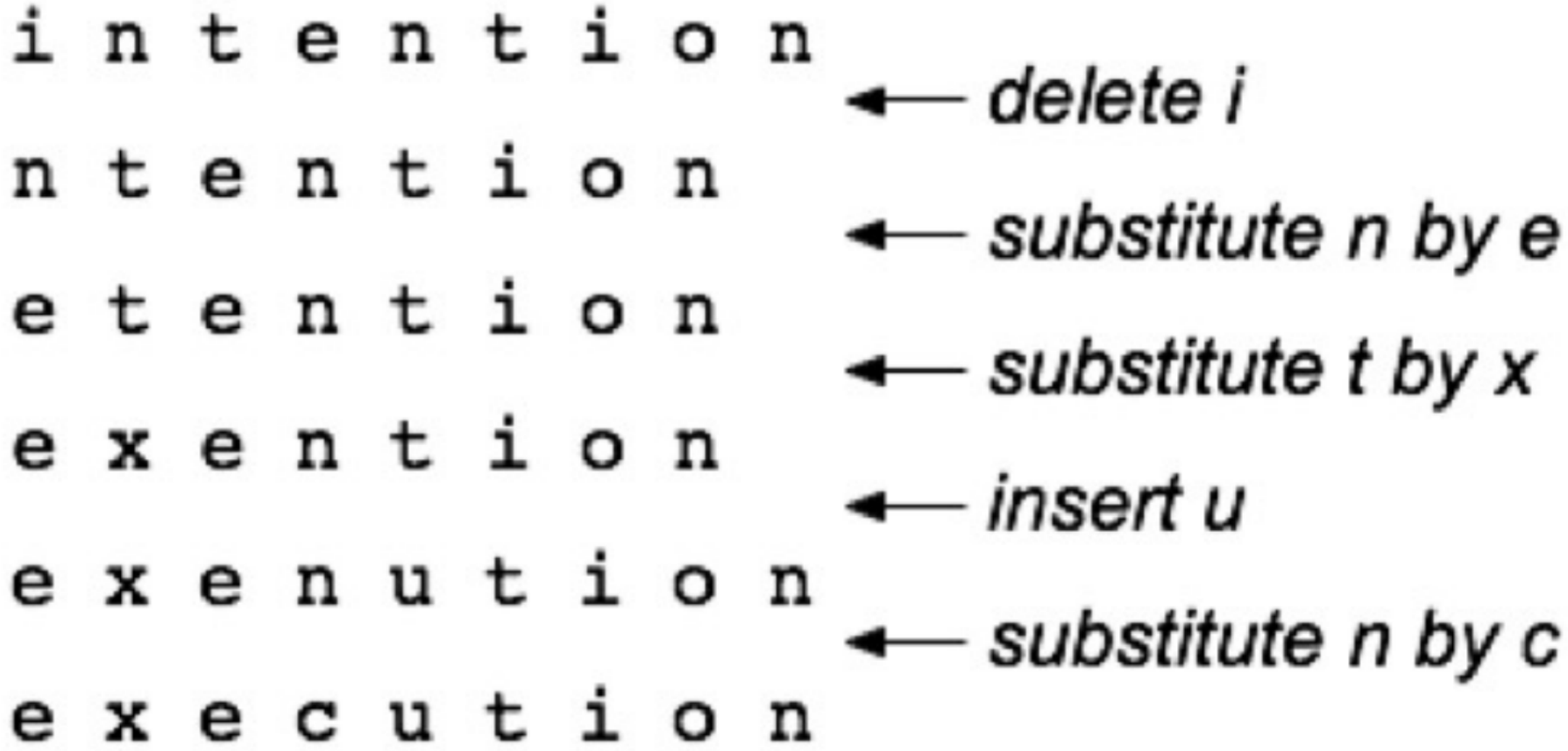
Edit Distance between two string
Basic edit operations

- Insertion
- Deletion
- Substitution

```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N

d   s   s     d   s
```

Edit transcript



MED - Computation

MED between "kitten" and "sitting"

kitten → sitten (substitution of "s" for "k")

sitten → sittin (substitution of "i" for "e")

sittin → sitting (insertion of "g" at the end).

MED: 3, LD: 5

Bottom-up dynamic programming

MED - Computation

Defining Min Edit Distance

- For two strings S_1 of len n, S_2 of len m
  - distance(i,j) or D(i,j)
    - means the edit distance of $S_1[1..i]$ and $S_2[1..j]$
    - i.e., the minimum number of edit operations need to transform the first i characters of S_1 into the first j characters of S_2
    - The edit distance of S_1, S_2 is D(n,m)
- We compute D(n,m) by computing D(i,j) for all i ($0 < i <$ n) and j ($0 <$ j $<$ m)

MED - Algorithm

- Initialization

```
D(i,0) = i
D(0,j) = j
```

- Recurrence Relation:

```
For each  i = 1…M
      For each  j = 1…N
                                          deletion
                                D(i-1,j) + 1   insertion
          D(i,j)= min{          D(i,j-1) + 1   substitution
                                D(i-1,j-1) +   2;  if X(i) ≠ Y(j)
                                               0;  if X(i) = Y(j)
```

- Termination:

```
D(N,M) is distance
```

MED - Algorithm

We have two words *a* and *b*

$a = a_1 \ldots a_n$ and $b = b_1 \ldots b_m$ is given by d_{mn}, defined by the recurrence

```
public static int minDistance(String word1, String word2)
{
    int len1 = word1.length();
    int len2 = word2.length();

    int[][] dp = new int[len1 + 1][len2 + 1];
    for (int i = 0; i <= len1; i++)
    {
          dp[i][0] = i;
    }
    for (int j = 0; j <= len2; j++)
    {
        dp[0][j] = j;
    }
```

```
for (int i = 0; i < len1; i++)
{
    char c1 = word1.charAt(i);
    for (int j = 0; j < len2; j++)
    {
        char c2 = word2.charAt(j);
        if (c1 == c2)
        {
            dp[i + 1][j + 1] = dp[i][j];
        }
        else
        {
            int replace = dp[i][j] + 1;
            int insert = dp[i][j + 1] + 1;
            int delete = dp[i + 1][j] + 1;

            int min = replace > insert ? insert : replace;
            min = delete > min ? min : delete;
            dp[i + 1][j + 1] = min;
        }
    }
}
return dp[len1][len2];
}
```

transform

2

i (0 < *i* <

N	9	
O	8	
I	7	
T	6	
N	5	
E	4	
T	3	
N	2	
I	1	
#	**0**	1
	#	E

N	9
O	8
I	7

MED- Tabular Visualization

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1									
#	**0**	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

MED- Tabular Visualization

N	9									
O	8									
I	7									
T	6									
N	5									
E	4									
T	3									
N	2									
I	1									
#	**0**	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 \\ D(i,j-1) + 1 \\ D(i-1,j-1) + \begin{cases} 2; & \text{if } S_1(i) \neq S_2(j) \\ 0; & \text{if } S_1(i) = S_2(j) \end{cases} \end{cases}$$

MED- Tabular Visualization

N	9	8	9	10	11	12	11	10	9	**8**
O	8	7	8	9	10	11	10	9	8	9
I	7	6	7	8	9	10	9	8	9	10
T	6	5	6	7	8	9	8	9	10	11
N	5	4	5	6	7	8	9	10	11	10
E	4	3	4	5	6	7	8	9	10	9
T	3	4	5	6	7	8	7	8	9	8
N	2	3	4	5	6	7	8	7	8	7
I	1	2	3	4	5	6	7	6	7	8
#	**0**	1	2	3	4	5	6	7	8	9
	#	E	X	E	C	U	T	I	O	N

MED- Tabular Visualization

elephant & relevant

		E	L	E	P	H	A	N	T
	0	1	2	3	4	5	6	7	8
R	1	1	2	3	4	5	6	7	8
E	2	1	2	2	3	4	5	6	7
L	3	2	1	2	3	4	5	6	7
E	4	3	2	1	2	3	4	5	6
V	5	4	3	2	2	3	4	5	6
A	6	5	4	3	3	3	3	4	5
N	7	6	5	4	4	4	4	3	4
T	8	7	6	5	5	5	5	4	3

Performance: time: O(*nm*) space: O(*nm*)

MED - Backtrace

- Edit distance isn't sufficient
  - We often need to **align** each character of the two strings to each other
- We do this by keeping a "backtrace"
- Every time we enter a cell, remember where we came from
- When we reach the end,
  - Trace back the path from the upper right corner to read off the alignment

MED - Backtrace

n	9	↓ 8	↙←↓ 9	↙←↓ 10	↙←↓ 11	↙←↓ 12	↓ 11	↓ 10	↓ 9	↙ **8**	
o	8	↓ 7	↙←↓ 8	↙←↓ 9	↙←↓ 10	↙←↓ 11	↓ 10	↓ 9	↙ **8**	← 9	
i	7	↓ 6	↙←↓ 7	↙←↓ 8	↙←↓ 9	↙←↓ 10	↓ 9	↙ **8**	← 9	← 10	
t	6	↓ 5	↙←↓ 6	↙←↓ 7	↙←↓ 8	↙←↓ 9	↙ **8**	← 9	← 10	←↓ 11	
n	5	↓ 4	↙←↓ 5	↙←↓ 6	↙←↓ 7	↙←↓ **8**	↙←↓ 9	↙←↓ 10	↙←↓ 11	↙↓ 10	
e	4	↙ 3	← 4	↙← **5**	← **6**	← 7	←↓ 8	↙←↓ 9	↙←↓ 10	↓ 9	
t	3	↙←↓ 4	↙←↓ **5**	↙←↓ 6	↙←↓ 7	↙←↓ 8	↙ 7	←↓ 8	↙←↓ 9	↓ 8	
n	2	↙←↓ **3**	↙←↓ 4	↙←↓ 5	↙←↓ 6	↙←↓ 7	↙←↓ 8	↓ 7	↙←↓ 8	↙ 7	
i	**1**	↙←↓ 2	↙←↓ 3	↙←↓ 4	↙←↓ 5	↙←↓ 6	↙←↓ 7	↙ 6	← 7	← 8	
#	**0**	1	2	3	4	5	6	7	8	9	
	#	**e**	**x**	**e**	**c**	**u**	**t**	**i**	**o**	**n**	

MED - Backtrace

- Base conditions:

  `D(i,0) = i` `D(0,j) = j`

  Termination:

  `D(N,M) is distance`

- Recurrence Relation:

```
For each  i = 1…M
      For each  j = 1…N
                          D(i-1,j) + 1
D(i,j)= min               D(i,j-1) + 1
                          D(i-1,j-1) +  2; if X(i) ≠ Y(j)
                                        0; if X(i) = Y(j)
```

deletion

insertion

substitution

```
                  LEFT
ptr(i,j)=         DOWN
                  DIAG
```

insertion

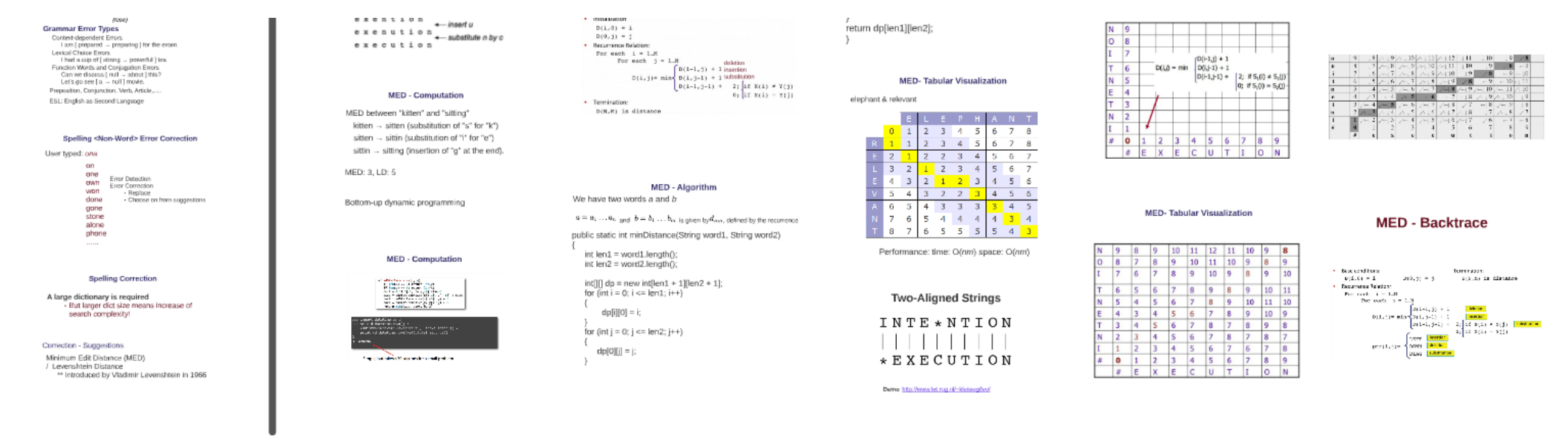
deletion

substitution

Performance: time: O(*nm*) space: O(*nm*)

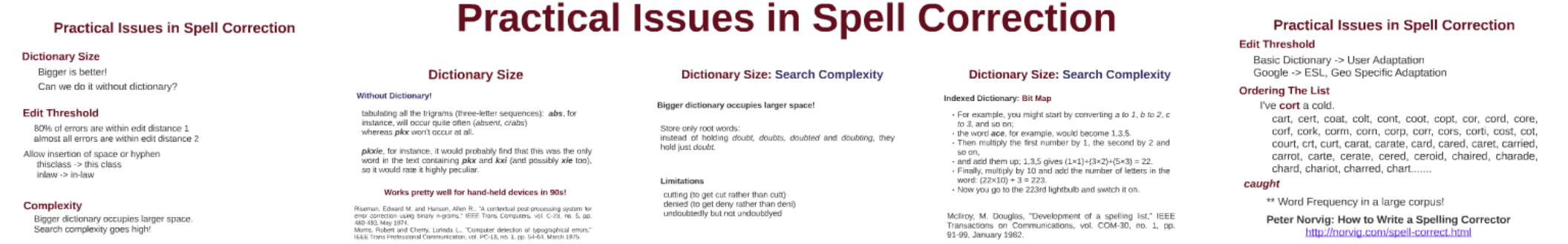
Two-Aligned Strings

Demo http://www.let.rug.nl/~kleiweg/lev/



Weighted Edit Distance

Practical Issues in Spell Correction



String Matching Paraphrase

Weighted MED

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	103	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	1	1	9	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

How these weightages have been calculated?

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

right: wrong1, wrong2

raining: rainning, raning
writings: writtings
disparagingly: disparingly
yellow: yello
four: forer, fours, fuore, fore*5, for*4
woods: woodes
hanging: haing
aggression: agression
eligible: eligble, elegable
electricity: electrisity, electricty*2
scold: schold, skold
adaptable: adabtable
caned: canned, cained
immature: imature
shouldn't: shoudln, shouldnt

e|i 917
a|e 856
i|e 771
e|a 749
a|i 559
t|te 478
r|re 392
s|c 383
e|ea 354
a|o 353
o|a 352
a|al 352
i|a 313
re|r 299
e|o 295
ea|e 285
te|t 271

Spelling Error - Reasons

ESL - speakers

Keyboard

Weighted MED - Revised Algorithm

- Initialization:

$$D(0,0) = 0$$

$$D(i,0) = D(i-1,0) + del[x(i)]; \qquad 1 < i \leq N$$

$$D(0,j) = D(0,j-1) + ins[y(j)]; \qquad 1 < j \leq M$$

- Recurrence Relation:

$$D(i,j) = \min \begin{cases} D(i-1,j) + del[x(i)] \\ D(i,j-1) + ins[y(j)] \\ D(i-1,j-1) + sub[x(i),y(j)] \end{cases}$$

- Termination:

$D(N,M)$ is distance

```
int w_med(String str1, String str2, int insert, int delete, int
substitute)
{
        int [][]distance = new int[str1.length+1][str2.length
+1];

        for(int i = 0; i <= str1.length; i++)
            distance[i][0] = i * delete;
        for(int j = 0; j <= str2.length; j++)
            distance[0][j] = j * insert;
        for(int i = 1; i <= str1.length; i++)
        {
            for(int j = 1; j <= str2.length; j++)
            {
                distance[i][j]= minimum(distance[i-1][j] +
delete,
                distance[i][j-1] + insert,
                distance[i-1][j-1] + ((str1[i-1] == str2[j-1])
                 ? 0 : substitute));
            }
        }
        return distance[str1.length][str2.length];
 }
```

Correction - Suggestions

Minimum Edit Distance (MED)
/ Levenshtein Distance
** Introduced by Vladimir Levenshtein in 1966

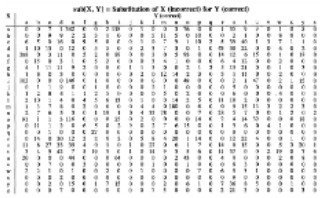
```
{
    dp[0][j] = j;
}
```

Demo http://www.let.rug.nl/~kleiweg/lev/

	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	1	2	3	4	5	6	7	8	9
3	3	2	1	2	3	4	5	6	7	8
4	4	3	2	1	2	3	4	5	6	7
5	5	4	3	2	1	2	3	4	5	6
6	6	5	4	3	2	1	2	3	4	5
7	7	6	5	4	3	2	1	2	3	4
8	8	7	6	5	4	3	2	1	2	3
9	9	8	7	6	5	4	3	2	1	2
10	10	9	8	7	6	5	4	3	2	1

Weighted Edit Distance

Weighted MED



How these weightages have been calculated?

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

right: wrong1, wrong2

raining: rainning, raning
writings: writtings
disparagingly: disparingly
yellow: yello
four: forer, fours, fuore, fore*5, for*4
woods: woodes
hanging: haing
aggression: agression
eligible: eligble, elegable
electricity: electrisity, electricty*2
scold: schold, skold
adaptable: adabtable
caned: canned, cained
immature: imalure
shouldn't: shoudln, shouldnt

e|i 917
a|e 856
i|e 771
e|a 749
a|i 559
t|te 478
r|re 392
s|c 383
e|ea 354
a|o 353
e|a 352
a|ai 352
i|a 313
re|r 299
e|o 295
ea|e 285
te|t 271

Spelling Error - Reasons

ESL - speakers

Keyboard

Weighted MED - Revised Algorithm

- Initialization:

$$D(0,0) = 0$$
$$D(i,0) = D(i-1,0) + del[x(i)]; \quad 1 < i \leq N$$
$$D(0,j) = D(0,j-1) + ins[y(j)]; \quad 1 < j \leq M$$

- Recurrence Relation:

$$D(i,j) = \min \begin{cases} D(i-1,j) & + del[x(i)] \\ D(i,j-1) & + ins[y(j)] \\ D(i-1,j-1) & + sub[x(i),y(j)] \end{cases}$$

- Termination:

$$D(N,M) \text{ is distance}$$

```
int w_med(String str1, String str2, int insert, int delete, int
substitute)
{
        int [][]distance = new int[str1.length+1][str2.length
+1];

        for(int i = 0; i <= str1.length; i++)
            distance[i][0] = i * delete;
        for(int j = 0; j <= str2.length; j++)
            distance[0][j] = j * insert;
        for(int i = 1; i <= str1.length; i++)
        {
            for(int j = 1; j <= str2.length; j++)
            {
                distance[i][j]= minimum(distance[i-1][j] +
delete,

                distance[i][j-1] + insert,
                distance[i-1][j-1] + ((str1[i-1] == str2[j-1])
                 ? 0 : substitute));
            }
        }
        return distance[str1.length][str2.length];
}
```

Practical Issues in Spell Correction

Practical Issues in Spell Correction

Dictionary Size

Bigger is better!

Can we do it without dictionary?

Edit Threshold

80% of errors are within edit distance 1
almost all errors are within edit distance 2

Allow insertion of space or hyphen
thisclass -> this class
inlaw -> in-law

Complexity

Bigger dictionary occupies larger space.
Search complexity goes high!

Dictionary Size

Without Dictionary!

tabulating all the trigrams (three-letter sequences): ***abs***, for instance, will occur quite often (*absent, crabs*) whereas ***plox*** won't occur at all.

plo**xie*, for instance, it would probably find that this was the only word in the text containing ***plox and ***lox*** (and possibly ***xie*** too), so it would rate it highly peculiar.

Works pretty well for hand-held devices in 90s!

Riseman, Edward M. and Hanson, Allen R., "A contextual post-processing system for error correction using binary n-grams," IEEE Trans Computers, vol. C-23, no. 5, pp. 480-493, May 1974.
Morris, Robert and Cherry, Lorinda L., "Computer detection of typographical errors," IEEE Trans Professional Communication, vol. PC-18, no. 1, pp. 54-64, March 1975.

Dictionary Size: Search Complexity

Bigger dictionary occupies larger space!

Store only root words:
instead of holding *doubt, doubts, doubted* and *doubting*, they hold just *doubt*.

Limitations

cutting (to get cut rather than cutt)
denied (to get deny rather than deni)
undoubtedly but not undoubtlyed

Dictionary Size: Search Complexity

Indexed Dictionary: Bit Map

- For example, you might start by converting *a to 1, b to 2, c to 3*, and so on;
- the word **ace**, for example, would become 1,3,5.
- Then multiply the first number by 1, the second by 2 and so on,
- and add them up: 1,3,5 gives (1×1)+(3×2)+(5×3) = *22*.
- Finally, multiply by 10 and add the number of letters in the word: (22×10) + 3 = 223.
- Now you go to the 223rd lightbulb and switch it on.

McIlroy, M. Douglas, "Development of a spelling list," IEEE Transactions on Communications, vol. COM-30, no. 1, pp. 91-99, January 1982.

Practical Issues in Spell Correction

Edit Threshold

Basic Dictionary -> User Adaptation
Google -> ESL, Geo Specific Adaptation

Ordering The List

I've **cort** a cold.

cart, cert, coat, colt, cont, coot, copt, cor, cord, core, corf, cork, corm, corn, corp, corr, cors, corti, cost, cot, court, crt, curt, carat, carate, card, cared, caret, carried, carrot, carte, cerate, cered, ceroid, chaired, charade, chard, chariot, charred, chart.....

caught

** Word Frequency in a large corpus!

Peter Norvig: How to Write a Spelling Corrector
http://norvig.com/spell-correct.html

String Matching Paraphrase

String Matching: Pre-Requisite for Many Applications

Original: Giraffes like Acacia leaves and hay and they can consume 75 pounds of food a day.

Paraphrase: A giraffe can eat up to 75 pounds of Acacia leaves and hay everyday.

MED?

Cosine Similarity

Vector Spaced Model

Cosine similarity is a measure of similarity between two vectors of an inner product space that measures the cosine of the angle between them.

$$a \cdot b = \|a\| \|b\| \cos\theta$$

Given two vectors of attributes, A and B, the cosine similarity, cos(θ), is represented using a dot product and magnitude

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

dotProduct = DotProduct(vecA, vecB);
cosine= dotProduct/(magnitudeOfA*magnitudeOfB)

Cosine Similarity

Text 1: Julie loves me more than Linda loves me

Text 2: Jane likes me more than Julie loves me

me: 2 2, Julie: 1 1, likes: 0 1, loves: 2 1, Jane: 0 1, Linda: 1 0, than: 1 1, more: 1 1

a: [2, 1, 0, 2, 0, 1, 1, 1]
b: [2, 1, 1, 1, 1, 0, 1, 1]

The cosine of the angle between them is about 0.822.

Cosine Similarity: Document

	abandon	...	plane	...	kill	...	survive	...	people
$\overrightarrow{doc\ 1}$ =	(0.11	...	1.00	...	0.03	...	0.23	...	0.65)
$\overrightarrow{doc\ 2}$ =	(0.00	...	0.01	...	1.00	...	0.11	...	0.09)

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

$$\cos(\text{doc 1}, \text{doc 2}) = \frac{0.11 \times 0.00 + \cdots + 0.65 \times 0.09}{\sqrt{0.11^2 + \cdots + 0.65^2} \times \sqrt{0.00^2 + \cdots + 0.09^2}} \in [0,1]$$

Cosine: Limitations

Can not capture Synonyms
WordNet: https://wordnet.princeton.edu/
Appropriate Stemming has to be used!

Other Similarity Measures

Tanimoto coefficient
Dice coefficient
Hamming distance
Soergel distance

Practical Issues in Spell Correction

Dictionary Size

Bigger is better!

Can we do it without dictionary?

Edit Threshold

80% of errors are within edit distance 1
almost all errors are within edit distance 2

Allow insertion of space or hyphen
thisclass -> this class
inlaw -> in-law

Complexity

Bigger dictionary occupies larger space.
Search complexity goes high!

Dictionary Size

Without Dictionary!

tabulating all the trigrams (three-letter sequences): ***abs***, for instance, will occur quite often (*absent*, *crabs*)
whereas ***pkx*** won't occur at all.

pkxie, for instance, it would probably find that this was the only word in the text containing ***pkx*** and ***kxi*** (and possibly ***xie*** too), so it would rate it highly peculiar.

Works pretty well for hand-held devices in 90s!

Riseman, Edward M. and Hanson, Allen R., "A contextual post-processing system for error correction using binary n-grams," IEEE Trans Computers, vol. C-23, no. 5, pp. 480-493, May 1974.
Morris, Robert and Cherry, Lorinda L., "Computer detection of typographical errors," IEEE Trans Professional Communication, vol. PC-18, no. 1, pp. 54-64, March 1975.

Dictionary Size: Search Complexity

Bigger dictionary occupies larger space!

Store only root words:
instead of holding *doubt, doubts, doubted* and *doubting*, they hold just *doubt.*

Limitations

cutting (to get cut rather than cutt)
denied (to get deny rather than deni)
undoubtedly but not undoubtlyed

Dictionary Size: Search Complexity

Indexed Dictionary: Bit Map

- For example, you might start by converting *a to 1*, *b to 2*, *c to 3*, and so on;
- the word ***ace***, for example, would become 1,3,5.
- Then multiply the first number by 1, the second by 2 and so on,
- and add them up; 1,3,5 gives (1×1)+(3×2)+(5×3) = 22.
- Finally, multiply by 10 and add the number of letters in the word: (22×10) + 3 = 223.
- Now you go to the 223rd lightbulb and switch it on.

McIlroy, M. Douglas, "Development of a spelling list," IEEE Transactions on Communications, vol. COM-30, no. 1, pp. 91-99, January 1982.

Practical Issues in Spell Correction

Edit Threshold

Basic Dictionary -> User Adaptation
Google -> ESL, Geo Specific Adaptation

Ordering The List

I've **cort** a cold.

cart, cert, coat, colt, cont, coot, copt, cor, cord, core, corf, cork, corm, corn, corp, corr, cors, corti, cost, cot, court, crt, curt, carat, carate, card, cared, caret, carried, carrot, carte, cerate, cered, ceroid, chaired, charade, chard, chariot, charred, chart.....

caught

** Word Frequency in a large corpus!

Peter Norvig: How to Write a Spelling Corrector
http://norvig.com/spell-correct.html

electricity: electinsity, electricity^2
scold: schold, skold
adaptable: adabtable
caned: canned, cained
immature: imature
shouldn't: shoudln, shouldnt

a|a| 352
i|a 313
re|r 299
e|o 295
ea|e 285
te|t 271

- Termination:
  D(N,M) is distance

```
                ? 0 : substitute));
            }
        }
        return distance[str1.length][str2.length];
    }
```

Practical Issues in Spell Correction

Dictionary Size

Bigger is better!

Can we do it without dictionary?

Edit Threshold

80% of errors are within edit distance 1
almost all errors are within edit distance 2

Allow insertion of space or hyphen
thisclass -> this class
inlaw -> in-law

Complexity

Bigger dictionary occupies larger space.
Search complexity goes high!

Practical Issues in Spell Correction

Dictionary Size

Without Dictionary!

tabulating all the trigrams (three-letter sequences): ***abs***, for instance, will occur quite often (*absent, crabs*) whereas ***pkx*** won't occur at all.

pkxie, for instance, it would probably find that this was the only word in the text containing ***pkx*** and ***kxl*** (and possibly ***xle*** too), so it would rate it highly peculiar.

Works pretty well for hand-held devices in 90s!

Riseman, Edward M. and Hanson, Allen R., "A contextual post-processing system for error correction using binary n-grams," IEEE Trans Computers, vol. C-23, no. 5, pp. 480-493, May 1974.
Mitton, Robert and Cherry, Lorinda L., "Computer detection of typographical errors," IEEE Trans Professional Communication, vol. PC-15, no. 1, pp. 54-64, March 1975.

Dictionary Size: Search Complexity

Bigger dictionary occupies larger space!

Store only root words:
instead of holding *doubt, doubts, doubted* and *doubting*, they hold just *doubt*.

Limitations

cutting (to get cut rather than cutt)
denied (to get deny rather than deni)
undoubtedly but not undoubtlyed

Dictionary Size: Search Complexity

Indexed Dictionary: Bit Map

- For example, you might start by converting *a to 1, b to 2, c to 3*, and so on;
- the word **ace**, for example, would become 1,3,5.
- Then multiply the first number by 1, the second by 2 and so on,
- and add them up; 1,3,5 gives (1×1)+(3×2)+(5×3) = 22.
- Finally, multiply by 10 and add the number of letters in the word: (22×10) + 3 = 223.
- Now you go to the 223rd lightbulb and switch it on.

McIlroy, M. Douglas, "Development of a spelling list," IEEE Transactions on Communications, vol. COM-30, no. 1, pp. 91-99, January 1982.

Practical Issues in Spell Correction

Edit Threshold

Basic Dictionary -> User Adaptation
Google -> ESL, Geo Specific Adaptation

Ordering The List

I've **cort** a cold.

cart, cert, coat, colt, cont, coot, copt, cor, cord, core, corf, cork, corm, corn, corp, corr, cors, corti, cost, cot, court, crt, curt, carat, carate, card, cared, caret, carried, carrot, carte, cerate, cered, ceroid, chaired, charade, chard, chariot, charred, chart.....

caught

** Word Frequency in a large corpus!

Peter Norvig: How to Write a Spelling Corrector
http://norvig.com/spell-correct.html

String Matching Paraphrase

String Matching: Pre-Requisite for Many Applications

Original: Giraffes like Acacia leaves and hay and they can consume 75 pounds of food a day.

Paraphrase: A giraffe can eat up to 75 pounds of Acacia leaves and hay everyday.

MED?

Cosine Similarity

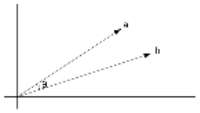
Vector Spaced Model

Cosine similarity is a measure of similarity between two vectors of an inner product space that measures the cosine of the angle between them.

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

dotProduct = DotProduct(vecA, vecB);
cosine= dotProduct/(magnitudeOfA*magnitudeOfB)

Cosine Similarity



Text 1: Julie loves me more than Linda loves me

Text 2: Jane likes me more than Julie loves me

me: 2 2, Julie: 1 1, likes: 0 1, loves: 2 1, Jane: 0 1, Linda: 1 0, than: 1 1, more: 1 1

a: [2, 1, 0, 2, 0, 1, 1, 1]
b: [2, 1, 1, 1, 1, 0, 1, 1]

The cosine of the angle between them is about 0.822.

Cosine Similarity: Document

	abandon	···	plane	···	kill	···	survive	···	people
$\overrightarrow{DOC1}$ =	(0.11	···	1.00	···	0.03	···	0.23	···	0.65)
$\overrightarrow{DOC2}$ =	(0.00	···	0.01	···	1.00	···	0.11	···	0.09)

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

$$\cos(\overrightarrow{DOC1}, \overrightarrow{DOC2}) = \frac{0.11 \times 0.00 + \cdots + 0.65 \times 0.09}{\sqrt{0.11^2 + \cdots + 0.65^2} \times \sqrt{0.00^2 + \cdots + 0.09^2}} \in [0,1]$$

Cosine: Limitations

Can not capture Synonyms
WordNet: https://wordnet.princeton.edu/
Appropriate Stemming has to be used!

Other Similarity Measures

Tanimoto coefficient
Dice coefficient
Hamming distance
Soergel distance

String Matching: Pre-Requisite for Many Applications

Original: Giraffes like Acacia leaves and hay and they can consume 75 pounds of food a day.

Paraphrase: A giraffe can eat up to 75 pounds of Acacia leaves and hay everyday.

MED?

Cosine Similarity

Vector Spaced Model

Cosine similarity is a measure of similarity between two vectors of an inner product space that measures the cosine of the angle between them.

The cosine of two vectors can be derived by using the Euclidean dot product formula:

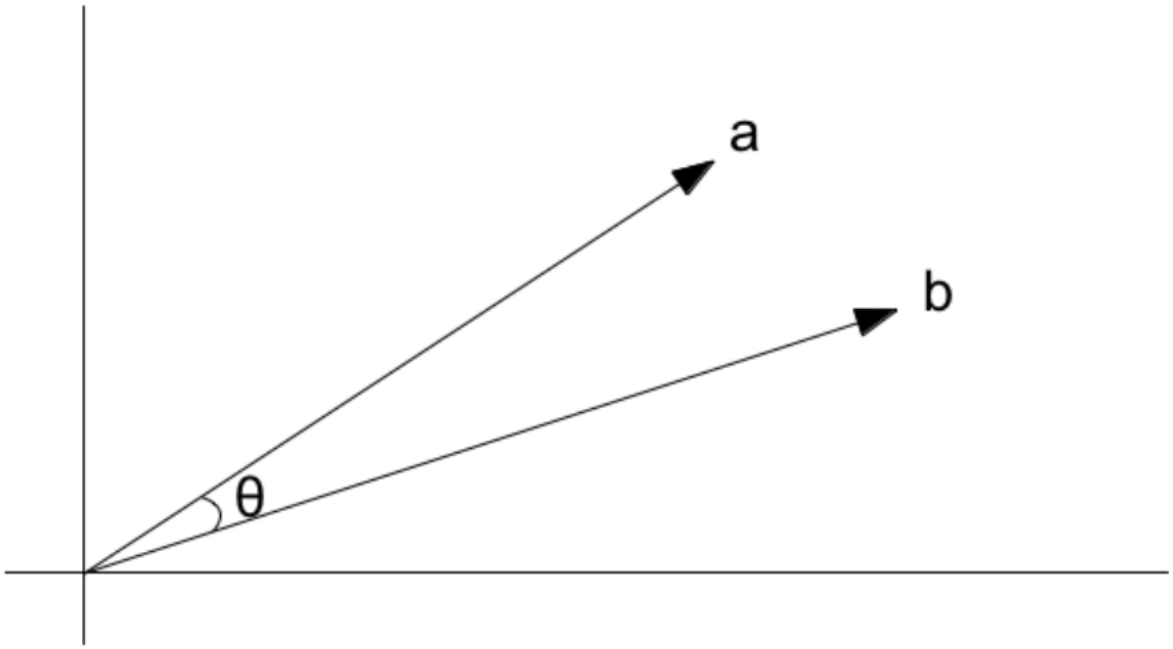
$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \, \|\mathbf{b}\| \cos\theta$$

Given two vectors of attributes, *A* and *B*, the cosine similarity, *cos(θ)*, is represented using a dot product and magnitude as

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

dotProduct = DotProduct(vecA, vecB);
cosine= dotProduct/(magnitudeOfA*magnitudeOfB)

Cosine Similarity



Text 1: Julie loves me more than Linda loves me

Text 2: Jane likes me more than Julie loves me

me: 2 2, Julie: 1 1, likes: 0 1, loves: 2 1, Jane: 0 1, Linda: 1 0, than: 1 1, more: 1 1

a: [2, 1, 0, 2, 0, 1, 1, 1]
b: [2, 1, 1, 1, 1, 0, 1, 1]

The cosine of the angle between them is about 0.822.

Cosine Similarity: Document

	abandon	⋯	plane	⋯	kill	⋯	survive	⋯	people
$\overrightarrow{\text{DOC 1}} =$	(0.11	⋯	1.00	⋯	0.03	⋯	0.23	⋯	0.65)
⋮									
$\overrightarrow{\text{DOC 2}} =$	(0.00	⋯	0.01	⋯	1.00	⋯	0.11	⋯	0.09)

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

$$sim(\text{DOC1 \& DOC2}) = \frac{0.11 * 0.00 + \cdots + 0.65 * 0.09}{\sqrt{0.11^2 + \cdots + 0.65^2} * \sqrt{0.00^2 + \cdots + 0.09^2}} \in [0,1]$$

Cosine: Limitations

Can not capture Synonyms

WordNet: https://wordnet.princeton.edu/

Appropriate Stemming has to be used!

Other Similarity Measures

Tanimoto coefficient

Dice coefficient

Hamming distance

Soergel distance