

Spelling and Grammar Correction

A screenshot of Microsoft Word's ribbon interface titled "Document1 - Microsoft Word". The "Home" tab is selected. The ribbon includes tabs for Home, Insert, Page Layout, References, Mailings, Review, View, Developer, and Acrobat. Below the ribbon are four main toolbars: Clipboard, Font, Paragraph, and Styles. The Font toolbar shows "Courier New" as the font and "10" as the size. The Paragraph toolbar has "List" selected. The Styles toolbar shows "Quick Styles" and "Change Styles". The main document area contains two paragraphs of text. The first paragraph reads: "I typed this sentence directly into Word to see if my misspellings would be underlined." The word "misspellings" is underlined with a red wavy line, indicating it is a misspelling. The second paragraph reads: "I typed this sentence in VS and cut and paste it into word to see if my misspellings would be underlined." The word "misspellings" is also underlined with a red wavy line in this paragraph. At the bottom of the screen, the status bar displays "Words: 36", the language "English (United States)", zoom controls at "100%", and other document information.

I typed this sentence directly into Word to see if my misspellings would be underlined.

I typed this sentence in VS and cut and paste it into word to see if my misspellings would be underlined.

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

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	← <i>delete i</i>
n t e n t i o n	← <i>substitute n by e</i>
e t e n t i o n	← <i>substitute t by x</i>
e x e n t i o n	← <i>insert u</i>
e x e n u t i o n	← <i>substitute n by c</i>
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

Defining Min Edit Distance

- For two strings S_1 of len n , S_2 of len m
 - $\text{distance}(i,j)$ 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 $i = 1 \dots M$

For each $j = 1 \dots N$

$$D(i, j) = \min \begin{cases} D(i-1, j) + 1 & \text{deletion} \\ D(i, j-1) + 1 & \text{insertion} \\ D(i-1, j-1) + 2; & \text{substitution} \\ 0; & \text{if } X(i) \neq Y(j) \\ & \text{if } X(i) = Y(j) \end{cases}$$

- Termination:

$D(N, M)$ is distance

MED - Algorithm

We have two words a and b

$a = a_1 \dots a_n$ and $b = b_1 \dots b_m$ 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;
    }
```

transform
2
| i (0 < i <

```
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];
}
```

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

N	9
O	8
T	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	

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
E	2	1	2	2	3	4	5	6
L	3	2	1	2	3	4	5	6
E	4	3	2	1	2	3	4	5
V	5	4	3	2	2	3	4	5
A	6	5	4	3	3	3	3	4
N	7	6	5	4	4	4	3	4
T	8	7	6	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:

$$D(i, 0) = i \quad D(0, j) = j$$

- Recurrence Relation:

For each $i = 1 \dots M$

For each $j = 1 \dots N$

$$D(i, j) = \min \begin{cases} D(i-1, j) + 1 & \text{deletion} \\ D(i, j-1) + 1 & \text{insertion} \\ D(i-1, j-1) + 2; & \begin{cases} \text{if } X(i) \neq Y(j) & \text{substitution} \\ 0; & \text{if } X(i) = Y(j) \end{cases} \\ \end{cases}$$
$$\text{ptr}(i, j) = \begin{cases} \text{LEFT} & \text{insertion} \\ \text{DOWN} & \text{deletion} \\ \text{DIAG} & \text{substitution} \end{cases}$$

Termination:

$D(N, M)$ is distance

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/>

Grammar Error Types
 Context-dependent Errors
 I am [prepared → preparing] for the exam.
 Let's go [go → going] ...
 I have off [leaving → leaving] ... powerful tea.
 Function Words and Conjugation Errors
 Don't we have [have → had] time for this?
 Let's see [see → will] more.
 Preposition, Conjunction, Verb, Article....
 ESL: English as Second Language

Spelling <Non-Word> Error Correction

User typed: one
 on
 one
 own
 Error Detection
 word
 - spell
 - Replace
 Choose on from suggestions
 gone
 stone
 alone
 phone

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

execute → insert u
 execution → substitute n by c

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



initialization:
 $D(0,0) = 0$
 $D(0,j) = j$

Recurrence Relation:
 For each $i = 1 \dots m$
 For each $j = 1 \dots n$
 deletion: $D(i,j) = \min(D(i-1,j) + 1, D(i,j-1) + 1)$
 substitution: $D(i,j) = \min(D(i-1,j-1) + 1, \max(D(i-1,j), D(i,j-1)) + 1)$
 $t_1[i][x(i)] = t_1[j]$
 $t_2[i][y(j)] = t_2[j]$

Termination:

$D(m,n)$ is distance

MED - Algorithm

We have two words a and b

$a = a_1 \dots a_m$ and $b = b_1 \dots b_n$ is given by d_{min} , 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;  

    }
```

return $dp[m][n]$;

MED - Tabular Visualization

elephant & relevant

	E	L	A	N	T					
R	0	1	2	3	4	5	6	7	8	9
I	1	1	2	3	4	5	6	7	8	9
E	2	1	2	3	4	5	6	7	8	9
A	3	2	1	2	3	4	5	6	7	8
N	4	3	2	1	2	3	4	5	6	7
T	5	4	3	2	1	2	3	4	5	6
O	6	5	4	3	2	1	2	3	4	5
N	7	6	5	4	3	2	1	2	3	4
E	8	7	6	5	4	3	2	1	2	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.csail.mit.edu/levenig.html>

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
S	4	3	4	5	6	7	8	9	10	9
E	3	4	5	6	7	8	9	8	9	8
A	2	3	4	5	6	7	8	7	8	7
L	1	2	3	4	5	6	7	6	7	8
E	0	1	2	3	4	5	6	7	8	9
X	1	2	3	4	5	6	7	8	9	8

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
S	4	3	4	5	6	7	8	9	10	9
E	3	4	5	6	7	8	9	10	9	8
A	2	3	4	5	6	7	8	9	10	9
L	1	2	3	4	5	6	7	6	7	8
E	0	1	2	3	4	5	6	7	8	9
X	1	2	3	4	5	6	7	8	9	8

MED - Backtrace

Initialization:	$d(0,0) = 0$	Recurrence Relation:	$d(i,j) = \min(d(i-1,j) + 1, d(i,j-1) + 1)$
Recurrence Relation:	$d(i,j) = \min(d(i-1,j) + 1, d(i,j-1) + 1)$	Backtrace:	$d(0,0) = 0$
Recurrence Relation:	$d(i,j) = \min(d(i-1,j) + 1, d(i,j-1) + 1)$	Backtrace:	$d(0,0) = 0$
Recurrence Relation:	$d(i,j) = \min(d(i-1,j) + 1, d(i,j-1) + 1)$	Backtrace:	$d(0,0) = 0$
Recurrence Relation:	$d(i,j) = \min(d(i-1,j) + 1, d(i,j-1) + 1)$	Backtrace:	$d(0,0) = 0$

```
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++){  

            if(str1.charAt(i) == str2.charAt(j)){  

                distance[i][j] = distance[i-1][j-1];  

            } else {  

                distance[i][j] = min(distance[i-1][j],  

                                     distance[i][j-1],  

                                     distance[i-1][j-1]) + 1;  

            }  

        }  

    }  

    return distance[str1.length][str2.length];  

}
```

Weighted MED - Revised Algorithm

- Initialization: $d(0,0) = 0$
- Recurrence Relation: $d(i,j) = \min(d(i-1,j) + w_i, d(i,j-1) + w_j, d(i-1,j-1) + w_{i,j})$
- Termination: $d(m,n)$ is distance

Weighted MED

Substitution of Y (correct)										
x	2	3	4	5	6	7	8	9	10	11
y	2	15	6	2	3	0	1	2	3	4
z	10	11	12	13	14	15	16	17	18	19
w	1	1	1	1	1	1	1	1	1	1
v	1	1	1	1	1	1	1	1	1	1
u	1	1	1	1	1	1	1	1	1	1
t	1	1	1	1	1	1	1	1	1	1
s	1	1	1	1	1	1	1	1	1	1
r	1	1	1	1	1	1	1	1	1	1
o	1	1	1	1	1	1	1	1	1	1
n	1	1	1	1	1	1	1	1	1	1
m	1	1	1	1	1	1	1	1	1	1
l	1	1	1	1	1	1	1	1	1	1
k	1	1	1	1	1	1	1	1	1	1
j	1	1	1	1	1	1	1	1	1	1
i	1	1	1	1	1	1	1	1	1	1
h	1	1	1	1	1	1	1	1	1	1
g	1	1	1	1	1	1	1	1	1	1
f	1	1	1	1	1	1	1	1	1	1
e	1	1	1	1	1	1	1	1	1	1
d	1	1	1	1	1	1	1	1	1	1
c	1	1	1	1	1	1	1	1	1	1
b	1	1	1	1	1	1	1	1	1	1
a	1	1	1	1	1	1	1	1	1	1



Dictionary Size

Without Dictionary

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

pxole, for instance, it would probably find that this was the only word in the text containing pxx and kxx (and possibly xie too), so it would rate it highly peculiar.

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

Risner, Edward M. and Hansen, Allen R., "A content-based post-processing system for error correction using trigrams," IEEE Trans. Computers, vol. C-32, no. 5, pp. 480-485, May 1983.
 Morris, Robert and Cherry, Lorinda L., "Computer detection of typographical errors," IEEE Trans Professional Communication, vol. PC-18, no. 3, pp. 14-16, 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 cut)
 denied (to get deny rather than den)
 undoubtedly but not undoubtably

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 $(1+1) \cdot (3 \cdot 2) \cdot (5 \cdot 3) = 22$.
- Finally, multiply by 10 and add the number of letters in the word: $(22 \cdot 10) + 3 = 223$.
- Now you go to the 223rd lightbulb and switch it on.

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

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	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	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	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	e i 917
raining: raining, raning	a e 856
writings: writtings	i e 771
disparagingly: disparingly	e a 749
yellow: yello	a i 559
four: forer, fours, fuore, fore*5, for*4	t te 478
woods: woodes	r re 392
hanging: haing	s c 383
aggression: agression	e ea 354
eligible: eligible, elegable	a o 353
electricity: electrisity, electricity*2	o a 352
scold: schold, skold	a al 352
adaptable: adaptable	i a 313
caned: canned, cained	re r 299
immature: imature	e o 295
shouldn't: shoudln, shouldnt	ea e 285
	te t 271

ESL - speakers

Keyboard

Spelling Error - Reasons



Weighted MED - Revised Algorithm

- Initialization:

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

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

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

- Recurrence Relation:

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

- 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];
}
```



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

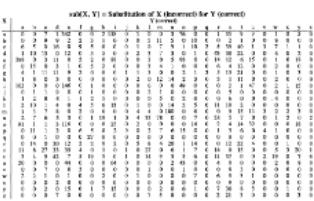
* E X E C U T I O N

N	Z	S	R	B	F	Y	T	S	R
1	1	2	3	4	5	6	7	6	7
#	0	1	2	3	4	5	6	7	8
#	E	X	E	C	U	T	I	O	N



Demo: <http://www.norvig.com/levenshtein.html>

Weighted MED



How these weightages have been calculated?

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

right: wrong1, wrong2
rainning: raining, raning
writings: writings
disparaginly: disparingly
yellow: yellow
four: fore, fours, flore, fore!5, for!4
woods: woodes
hanging: hang
agressions: agression
elptical: elliptical
electricity: electricity*2
scoid: schloid, skloid
adaptable: adaptable
canned: canned, cained
immature: immature
shouldn't: shoudnin, shouldin

ESL - speakers

Keyboard



Weighted MED - Revised Algorithm

- Initialization: $d(0,0) = 0$
- $d(i,0) = d(i-1,0) + del(x[i])$; $1 \leq i \leq n$
- $d(0,j) = d(0,j-1) + ins(y[j])$; $1 \leq j \leq m$
- 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];
    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] +
                                      distance[i][j-1] + insert,
                                      distance[i-1][j] + 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!

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.

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

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

Riesenman, Edward M. and Hanson, Allen P. "A contextual spell processing system for error correction using library trigrams," IEEE Trans Computers, vol. C-23, no. 5, pp. 480-493, May 1974.
Mehri, Rehbet and Cherry, Lorinda L., "Computer detection of typographical errors," IEEE Trans Professional Communication, vol. PC-33, no. 1, pp. 54-64, March 1990.

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 cut)
denied (to get deny rather than deni)
undoubtedly but not undoubtably

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 $(1 \times 1) + (3 \times 2) + (5 \times 3) = 22$.
- Finally, multiply by 10 and add the number of letters in the word: $(22 \times 10) + 3 = 223$.
- 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 **caught** a cold.
cart, cart, coat, colt, cont, coot, cop, cor, cord, core,
corp, cork, corn, corn, corp, corp, cors, cort, cost, cot,
court, crt, cur, 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

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.

$$\text{cosine} = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_i A_i B_i}{\sqrt{\sum_i A_i^2} \sqrt{\sum_i B_i^2}}$$

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]
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	---
(0.11	---	1.00	---	0.03	---	0.23	---	0.65)	---

dotProduct = $\frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$

similarity = cosine = $\frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$

cosine(A,B) = $\frac{A_1 B_1 + A_2 B_2 + \dots + A_n B_n}{\sqrt{A_1^2 + A_2^2 + \dots + A_n^2} \sqrt{B_1^2 + B_2^2 + \dots + B_n^2}}$

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 $(1 \times 1) + (3 \times 2) + (5 \times 3) = 22$.
- Finally, multiply by 10 and add the number of letters in the word: $(22 \times 10) + 3 = 223$.
- 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: 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:

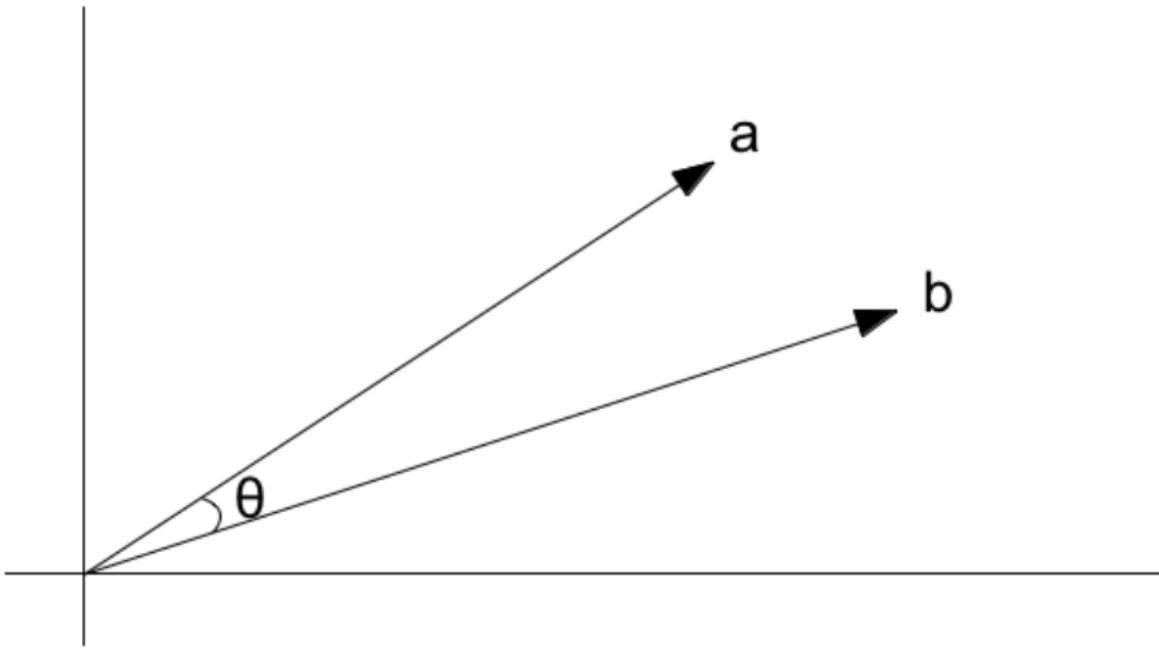
$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \cos \theta$$

Given two [vectors](#) of attributes, A and B , the cosine similarity, $\cos(\theta)$, 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)

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

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

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