PERFORMANCE AND QUALITY OF METHOD FOR SHORT TEXT SIMILARITY ALGORITHM BASED ON EDIT DISTANCE AND THESAURUS

Artur Niewiarowski, Marek Stanuszek
Cracow University of Technology, Cracow, Poland

About the Levenshtein distance

- The Levenshtein distance between two strings is equal to the minimum number of insertions, deletions and substitutions of chars required to change one string into the second one.
- The algorithm creates a matrix where its last element states as the solution.

Levenshtein distance algorithm is described by the formula:

$$\prod_{i=1}^{N}\prod_{j=1}^{M} d(i,j) = \min(d(i-1,j)+1, d(i,j-1)+1, d(i-1,j-1)+\beta)$$

$$\begin{cases} \beta = 0 : a(i) \equiv b(j) \\ \beta = 1 : a(i) \neq b(j) \\ d(i,0) = i \\ d(0,j) = j \\ d(0,0) = 0 \end{cases}$$

		K	U		K	D	M	'	1	3
	0	1	2	3	4	5	6	7	8	9
K	1	0	1	2	3	4	5	6	7	8
U	2	1	0	1	2	3	4	5	6	7
	3	2	1	0	1	2	3	4	5	6
K	4	3	2	1	0	1	2	3	4	5
D	5	4	3	2	1	0	1	2	3	4
M	6	5	4	3	2	1	0	1	2	3
'	7	6	5	4	3	2	1	0	1	2
1	8	7	6	5	4	3	2	1	0	1
4	9	8	7	6	5	4	3	2	1	1

Fig. 1. Example of Levenshtein matrix

where:
$\prod_{i=1}^{N}$ – symbol for the iteration, for i = (1, ..., N),

d – matrix sizes N+1, M+1, made from two terms,
N, M – length of two terms,
$d(i,j)$ - (i,j) – element of matrix **d**,
min – function returns minimum of two variables,
β – variable that gets values: 0 or 1,
a(i) – i – element in string of term a,
b(j) – j – element in string of term b.

Levenshtein distance K is a minimum number of operations (insertion, deletion, substitution) required to change one term into the other.

$K = d(N,M)$

Details of a problem:

- ❑ terms coding based on Levenshtein distance and thesaurus
- ❑ spelling mistakes in texts
- ❑ similarity measure based on edit distance

Used technologies:

- ❑ Microsoft .NET (Framework 4.0)
- ❑ Xamarin Mono (for OS Linux)

Examples of the use:

- ❑ texts (documents) analysis
- ❑ detecting plagiarism (in most cases - resignation of variety of nouns and verbs based on standard thesaurus)

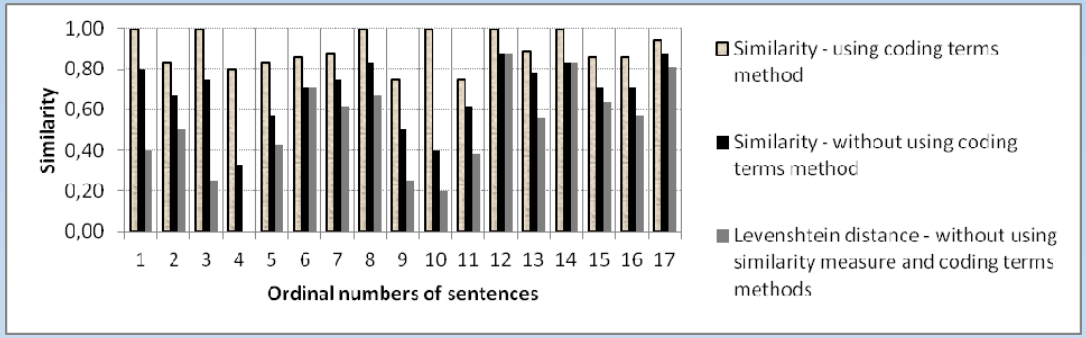

Fig. 2. Graphical results of quality test of English sentences. For all tests in this case acceptable boundaries of similarity P were: q=0.80 for thesaurus and q=0.75 for similarity between terms in sentences and qS =0.75 between sentences

Num. of sentences	Correct sentences	Incorrect sentences with synonyms
s1	Tom is writing a letter	**Dere** is **writin** a **letters**
s2	We are waiting for a taxi	We are **waitin** for **car**
s3	Is Mary having breakfast?	Is **Jane hasing brekfest**?
s4	Tom is not writing a letter	**Jimm** isn't **writin leter**
s5	He isn't looking at the stars	He **is not look** at the **start**
s6	He drinks milk twice a day	He **is** drinks **water** twice a day
s7	We go to work six times a week	We **goes** to **works seven** times a week
s8	I always feel great in spring	I **alway** feel great in **summer**
s9	Do you like apples?	**Does** you **likes pear**?
s10	I don't like milk	I **do not** likes water
s11	Tom was writing the letter all day yesterday	**Jimmy writting** the **leter** all day **yestaredy**

Num. of sentences	Correct sentences after terms coding	Incorrect sentences with synonyms after terms coding method
s1	#1 is writing a letter	#1 is writin a letters
s2	we are waiting for a #2	we are waitin for #2
s3	is #1 having breakfast?	is #1 hasing brekfest?
s4	#1 #9 writing a letter	#1 #9 writin leter
s5	he #9 looking at the stars	he #9 look at the start
s6	he drinks #12 twice a day	he is drinks #12 twice a day
s7	we go to work #3 times a week	we goes to works #3 times a week
s8	i always feel great in #4	i alway feel great in #4
s9	do you like #5?	does you likes #5?
s10	i #11 like #12	i #11 likes #12
s11	#1 was writing the letter all day yesterday	#1 writting the leter all day yestaredy

The obtained results show that the method of coding terms increases the precision of similarity estimation in some cases from 0-20% even up to 75-100%.