TECHNICAL TRANSACTIONS | **CZASOPISMO TECHNICZNE**
FUNDAMENTAL SCIENCES | NAUKI PODSTAWOWE
1-NP/2016

ARTUR NIEWIAROWSKI*

SHORT TEXT SIMILARITY ALGORITHM BASED ON THE EDIT DISTANCE AND THESAURUS

ALGORYTM PODOBIEŃSTWA KRÓTKICH FRAGMENTÓW TEKSTÓW OPARTY NA ODLEGŁOŚCI EDYCYJNEJ I SŁOWNIKU WYRAZÓW BLISKOZNACZNYCH

Abstract

This paper proposes a method of comparing the short texts using the Levenshtein distance algorithm and thesaurus for analysing terms enclosed in texts instead of popular methods exploiting the grammatical variations glossary. The tested texts contain a variety of nouns and verbs together with grammatical or orthographical mistakes. Based on the proposed new algorithm the similarity of such texts will be estimated. The described technique is compared with methods: Cosine distances, distance Dice and Jaccard distance constructed on the term frequency method. The proposition is competitive against well-known algorithms of stemming and lemmatization.

Keywords: *Levenshtein distance algorithm*, *the edit distance*, *thesaurus*, *the measure of texts similarity*, *plagiarism detection, text mining*, *Natural Language Processing*, *Natural Language Understanding*, *stemming*, *lemmatization*

Streszczenie

Artykuł przedstawia propozycję metody porównywania krótkich fragmentów tekstów bazującą na algorytmie odległości Levenshteina i słowniku wyrazów bliskoznacznych. Porównywane teksty zawierają odmienione terminy oraz celowe błędy ortograficzne i gramatyczne. Opisany mechanizm zestawiony został z popularnymi metodami porównywania tekstów, takimi jak: odległości Kosinusowa, Dice'a i Jaccard'a, dla których wartości wektorów obliczane są metodą częstości terminów. Zastosowanie w mechanizmie słownika wyrazów bliskoznacznych jest alternatywą wobec znanych algorytmów określania rdzenia terminu i lematyzacji w analizie danych tekstowych.

Słowa kluczowe: *odległość Levenshteina*, *odległość edycyjna*, *słownik wyrazów bliskoznacznych*, *miara podobieństwa tekstów*, *detekcja plagiatu*, *analiza danych tekstowych*, *przetwarzanie języka naturalnego*, *stemming*, *lematyzacja*

DOI: 10.4467/2353737XCT.16.149.5760

* Artur Niewiarowski (aniewiarowski@pk.edu.pl), Institute of Computer Science, Faculty of Physics, Mathematics and Computer Science of Cracow University of Technology.

1. Introduction

Most of the mechanisms estimating measures of similarity between text documents are based on vector space models and weight methods [1, 2, 3] with compilation of other methods, such as probability methods [4], semantic networks (e.g. WordNet) [5, 6] or genetic algorithms [7], etc. The traditional text identification mechanisms usually use glossary of grammatical variations which in some cases are difficult to implement. Most of the mechanisms are composed of stemming algorithms [8], such as popular: Lovins stemming algorithm [9], Porter stemming algorithm [10], Dawson stemming algorithm, Krovetz stemming algorithm, etc. [11], causing increasing of time consumed by identification algorithm during the data analysing process.

In our research we propose a model of effective mechanism for the calculation of similarity between two short texts (e.g. sentences) mainly based on Levenshtein distance algorithm (Lda) combined with the word coding technique. Our research indicates that the terms coding technique with its implementation for measure of text similarity improves the results of text identification significantly. The proposed technique seems to be easy for implementation in most programming technologies and open to most European languages.

2. Description of the problem

The main idea of the mechanism of measuring the similarity of texts consists of:

– implementation of function of terms coding based on Levenshtein distance [16] (*presented in subsection 2.2*) and thesaurus (*described in subsection 3.2*),
– calculation of similarity measure between two sentences based on Levenshtein distance between encoded terms.

2.1. Levenshtein distance algorithm

The concept of the Levenshtein distance algorithm (*Levenshtein Distance function*) may be depicted by the following pseudo-code:

Pseudo-code 1

```
input variables: char Text1[0..M-1], char Text2[0..N-1]
     declare: int d[0..M, 0..N]
     for i from 0 to M
      d[i, 0] := i
     for j from 0 to N
      d[0, j] := j

     for i from 1 to M
      for j from 1 to N

       if char of Text1 at (i - 1) = char of Text2 at (j - 1) then
                    cost := 0 else cost := 1
       end if
                    d[i, j] :=
                        Minimum(d[i - 1, j] + 1,
```

```
                    d[i, j - 1] + 1,
                    d[i - 1, j - 1] + cost)
      end for (variable j)
    end for (variable i)

return d[M, N];
```

where:

- d – Levenshtein matrix of the size N+1, M+1, formed for two terms: Text1 and Text2,
- M, N – lengths of two terms respectively,
- d [i, j] – (i, j) – element of Levenshtein matrix d,
- min – a function to calculate minimum of three variables,
- cost – variable that gets values either 0 or 1

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

$$K = d(M, N)$$

2.2. The measure of similarity between terms

Measure of similarity P is the quotient of number of Levenshtein operations (*after calculation of Lda*) by the number of all Levenshtein operations in pessimistic case. This means, before the calculations of Lda will be completed but with the maximum possible number of Levenshtein operations well known.

The similarity measures P is calculated by the formula:

$$P = 1 - \left(\frac{K}{K_{\max}} \right); \quad K_{\max} = \max(N, M), \quad \begin{matrix} K \geq 0,\ M > 0,\ N > 0 \\ P \in \langle 0,1 \rangle \end{matrix} \tag{1}$$

where:

- $K_{\max}$ – the length of the longest of analysed two terms/text strings (i.e. pessimistic case where K is equal to the length of the longest term).

Table 1

Examples of the Levenshtein distance and the measure of similarity between two short texts

No.	Text1	Text 2	*K*	*P*
1.	Car	Cars	1	0.75
2.	University	Universities	3	0.75
3.	Tom is writing a letter	Tom is writin letters	4	0.82

2.3. The algorithm to measure similarity between two sentences

The algorithm for measuring of similarity between two sentences, based on Lda, is described by the formula (2) presented below:

$$\mathop{\mathrm{I}}_{i_S=1}^{N_S}\mathop{\mathrm{I}}_{j_S=1}^{M_S} d_S(i_S, j_S) = \min\left(d_S(i_S-1, j_S)+1, d_S(i_S, j_S-1)+1, d_S(i_S-1, j_S-1)+\beta_S\right)$$

$$\begin{cases} \beta_S = 0 : \Lambda(a_S(i_S), b_S(j_S)) >= q \\ \beta_S = 1 : \Lambda(a_S(i_S), b_S(j_S)) < q \\ d_S(i_S, 0) = i_S \\ d_S(0, j_S) = j_S \\ d_S(0,0) = 0 \end{cases} \tag{2}$$

where:

I	–	symbol for iteration (for loop presented in the pseudo-code 1)
N_S+1, M_S+1	–	matrix sizes made from two sentences,
d_S	–	matrix made from two sentences,
$d_S(i_S,j_S) - (i_S,j_S)$	–	element of matrix d_S,
Λ	–	function which returns the measure of similarity between two terms P, calculated based on Levenshtein distance algorithm (pseudo-code 1), few terms creates sentence,
β_S	–	variable: 0 or 1,
$a_S(i_S)$– i_S	–	term of sentence a_S,
$b_S(j_S) - j_S$	–	term of sentence b_S
q	–	acceptable boundary of value of similarity measure for two texts (terms in this case). This value is set by the user and it depends on data (e.g. texts from old books, texts from newspapers).

The asymptotic computational complexity of the algorithm is $O(n^4)$. This derives from the construction of the algorithm which consists of four nested loops[1].

The similarity measures P_S between two sentences may be estimated in the rule:

$$P_S = 1 - \left(\frac{K_S}{K_{S\max}}\right); \quad K_{S\max} = \max(N_S, M_S), \quad \begin{matrix} K_S \geq 0,\ M_S > 0,\ N_S > 0 \\ P_S \in \langle 0,1 \rangle \end{matrix} \tag{3}$$

Table 2

Examples of the Levenshtein distance and the measure of similarity between two sentences in whose terms are treated as chars

No.	Sentence 1	Sentence 2	K_S	P_S	q
1.	My car isn't working	My bicycle isn't working	1	0,75	1; 0,75; 0,3
2.	What did you do yesterday?	What have you done?	3	0,40	1; 0,6
3.	What did you do yesterday?	What have you done?	3	0,60	0,5; 0,1
4.	Tom is writing a letter	Tom is writin letters	3	0,40	1; 0,9
5.	Tom is writing a letter	Tom is writin letters	3	0,80	0,85; 0,05

[1] Interesting research about parallelization of the Levenshtein distance algorithm (and Levenshtein--Damerau distance algorithm) for accelerate the calculations is described in [12].

3. Procedure of Terms Coding based on thesaurus

On the figures and tables below concept of the coding process is described. Tables 3–5 include sample data to be encoded. Formulas 4–7 describe all steps of the coding process.

3.1. The data model

The database model of thesaurus (τ) with tables of unique terms, groups of terms and table of terms associations is presented below.

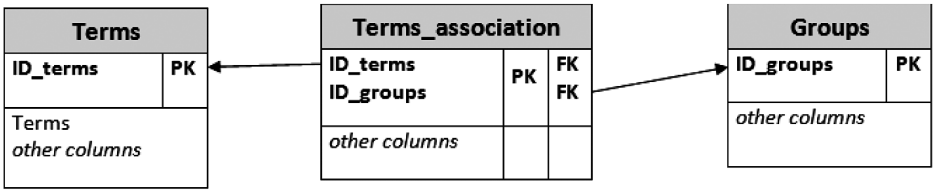

Fig. 1. The database model of thesaurus (τ)

Example of use of the model above:

Table 3

Terms

ID_terms	Terms
1	Tom
2	Mary
3	John
4	car
5	auto
6	vehicle

Table 4

Terms_association

ID_terms	ID_groups
1	1
2	1
3	1
1	2
2	2
4	4
5	5
6	6

Table 5

Groups

ID_groups	Describe
1	names
2	my best friends
3	vehicles

It is easy to see that a term can belong to a few groups. The example shows that *Tom* and *Mary* can belong to the groups: *names* and *my best friends*. It means that *Tom*, *Mary* and *John* are the same terms (names) because they have the same meaning. Following terms: *car, auto, vehicle* are the same terms – *vehicle*.

3.2. The coding process

The process of common group analysis proceeds with the following steps:

1. Get all codes of each terms of both texts (a_S, b_S) from thesaurus (e.g. array of codes for cases where one term can has a multiple meanings).

$$\Psi(a_s, \tau, q) \rightarrow \mathbf{C}_{\mathbf{a_S}}(i_S, t_{i_S}) \quad \text{and} \quad \Psi(b_s, \tau) \rightarrow \mathbf{C}_{\mathbf{b_S}}(j_S, t_{j_S}) \tag{4}$$

where:

Ψ – function to get codes of terms,

τ – thesaurus,

t_{i_S} – the number of term's codes variants,

$\mathbf{C}_{\mathbf{a_S}}$ – array of codes of terms of sentence a_S.

2. Calculate number of occurrences of codes terms based on their frequency in texts (i.e. matching process of common meanings).

$$\Gamma(\mathbf{C}_{\mathbf{a_S}}, \mathbf{C}_{\mathbf{b_S}}) \rightarrow \mathbf{C}_{\mathbf{ab_S}}(h) \tag{5}$$

where:

Γ – function which calculates frequency of occurrences of codes,

$\mathbf{C}_{\mathbf{ab_S}}$ – array of the best codes of terms,

h – ID of the term.

3. Replace each term in both texts with the most frequent code

$$\Phi(\mathbf{C}_{\mathbf{ab_S}}, \mathbf{C}_{\mathbf{a_S}}) \rightarrow \mathbf{NC}_{\mathbf{a_S}}(i_S) \quad \text{and} \quad \Phi(\mathbf{C}_{\mathbf{ab_S}}, \mathbf{C}_{\mathbf{b_S}}) \rightarrow \mathbf{NC}_{\mathbf{b_S}}(j_S) \tag{6}$$

where:

Φ – function which replaces the most frequent code,

$\mathbf{NC}_{\mathbf{b_S}}$ – new sentence with encoded terms.

4. Function which calculates similarity K_S between two sentences (short texts).

$$\Omega(\mathbf{NC}_{\mathbf{a_S}}, \mathbf{NC}_{\mathbf{b_S}}, q, q_\tau) \rightarrow P_S \tag{7}$$

where:

Ω – function which calculates similarity K_S measures between sentences,

q_τ – acceptable border value of similarity measure for two terms – between term includes in text and term derives from thesaurus.

4. Verification of proposed similarity measures mechanism

The following tests show how term coding methods improve the mechanism of similarity between two short texts. As a test of the proposed algorithms, 170 pars of correct and incorrect sentences written in various tenses were checked. 10% of interesting sentences were chosen and described below. Some examples based on the three popular languages of Eastern Europe are provided in Appendix 1.

The terms and sentences used for the tests were presented in the tables below:

1. synonyms (thesaurus) (Table 6)
2. grammatically and spelling correct sentences written in various tenses (Table 7)
3. grammatically and spelling incorrect sentences written based on the correct sentences (Table 7)
4. encoded texts based on the thesaurus (Table 8)
5. results of tests (Table 9)

For all tests and in all cases the acceptable boundaries of similarity P are: $q_\tau = 0.80$ for **thesaurus** and $q = 0.75$ for similarity between **terms** in sentences (formula 2).

Table 6

Example of the thesaurus schema with terms groups by common ID (code in text). Similar term from column *Describe* in analysing text will be replaced by ID

ID of groups	Describe (e.g. name of the group)	Terms
#1	names	Tom, Mary, John, Jimmy, Jane, Derek, Gina
#2	cars	car, auto, automobile, taxi, vehicle
#3	numbers	one, two three, four, five, six, seven, eight, nine, ten
#4	seasons	spring, summer, autumn, winter
#5	fruits	apple, pear, cherry, mango, kiwi, watermelon
#6	cities	Warsaw, Berlin, London
#7	phones	phone, telephone, iPhone, mobile phone
#8	very	very, extremely
#9	shortcuts1	is not, isn't
#10	shortcuts2	are not, aren't
#11	shortcuts3	don't, do not
#12	fluid	milk, water
#13	my_friends	Tom, Jack, Ella, Olivia

Table 7

Correct and incorrect sentences for the tests

No.	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
s12	They met when they were studying in Berlin	They met when they were studying in Warsaw
s13	I was working in London this time last year	I was work in Berlin this times last years

s14	I have found his telephone number	I have found his phone number
s15	I was shocked when I found out that Derek and Gina had got divorced	I was shock when I found out that John and Mary has gotten divorced
s16	I have been working for five hours	I has been working for six hour
s17	It had been raining for days so when they finally left, the roads were very muddy	It has been raining for days so when they finaly left, the roads were extremly muddy

Table 7 shows the correct sentences with synonyms in the left column and incorrect sentences with synonyms in the right column. Synonyms (not all) include mistakes, like for example Dere instead of Derek in the first row.

Table 8

Correct and incorrect sentences after terms coding (based on the thesaurus)

No.	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
s12	they met when they were studying in #6	they met when they were studying in #6
s13	i was working in #6 this time last year	i was work in #6 this times last years
s14	i have found his #7 number	i have found his #7 number
s15	i was shocked when i found out that #1 and #1 had got divorced	i was shock when i found out that #1 and #1 has gotten divorced
s16	i have been working for #3 hours	i has been working for #3 hour
s17	it had been raining for days so when they finally left, the roads were #8 muddy	it has been raining for days so when they finaly left, the roads were #8 muddy

Table 8 includes sentences from table 7 after coding by the proposed algorithm. The similarity of these sentences was calculated with (1)(2) and presented below.

Table 9

Values of similarity of the sentences with and without the terms coding method based on Levenshtein distance algorithm. Description of columns: Col. 1 – Similarity between correct and incorrect sentences without methods of the: similarity measure between terms; coding terms (based on Table 7 data); Col. 2 – Similarity between correct and incorrect sentences (without using method of terms coding) based on Table 7 data; Col. 3 – Similarity between correct and incorrect sentences (with using method of terms coding) based on Table 8 data; No – number of sentence

No.	Col. 1	Col. 2	Col. 3
s1	0.40	0.80	1.00
s2	0.50	0.67	0.83
s3	0.25	0.75	1.00
s4	0.00	0.33	0.80
s5	0.43	0.57	0.83
s6	0.71	0.71	0.86
s7	0.62	0.75	0.88
s8	0.67	0.83	1.00
s9	0.25	0.50	0.75
s10	0.20	0.40	1.00
s11	0.38	0.62	0.75
s12	0.88	0.88	1.00
s13	0.56	0.78	0.89
s14	0.83	0.83	1.00
s15	0.64	0.71	0.86
s16	0.57	0.71	0.86
s17	0.81	0.88	0.94

The obtained results show that the method of coding terms (column no. 3) increases the precision of similarity estimation in some cases from 0–20% even up to 75–100%.

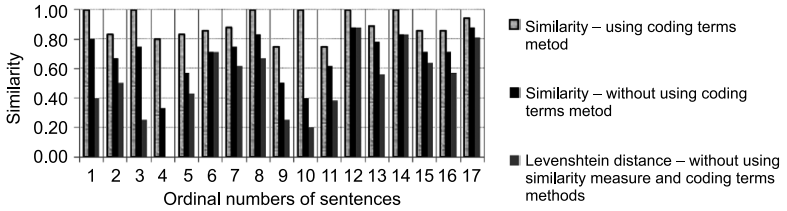

Fig. 2. Graphical results of quality test of English sentences. For all tests in this case acceptable boundaries of similarity P are: $q = 0.75$, $q_\tau = 0.80$

5. Comparison quality of described method with popular methods

Results in table 10 (and in tables 11–16 in Appendix 1) show that the similarity methods based on Levenshtein distance algorithm (i.e. Lda without coding terms method and Lda with coding terms method – Table 9/Col. 3) are more precise than popular methods like: Dice distance, Jaccard distance or Cosine distance [17]. Formulas (8–12) refer to these distances.

Dice distance is described by the formula below:

$$\text{Dice\_dist}(a_i, b_k) = 2\frac{\sum_{j=1}^{n} a_{ij} b_{kj}}{\sum_{j=1}^{n} a_{ij}^2 \sum_{j=1}^{n} b_{kj}^2} \tag{8}$$

where:

a_i – weight of the term in i position of the vector of text document a,

n – length of the two vectors created from two text documents a and b.

Jaccard distance is described by the formula below:

$$\text{Jaccard\_dist}(a_i, b_k) = \frac{\sum_{j=1}^{n} a_{ij} b_{kj}}{\sum_{j=1}^{n} a_{ij}^2 + \sum_{j=1}^{n} b_{kj}^2 - \sum_{j=1}^{n} a_{ij} b_{kj}} \tag{9}$$

Cosine distance is described by the formula below:

$$\text{Cosine\_dist}(a_i, b_k) = \frac{\sum_{j=1}^{n} a_{ij} b_{kj}}{\sqrt{\sum_{j=1}^{n} a_{ij}^2} \sqrt{\sum_{j=1}^{n} b_{kj}^2}} \tag{10}$$

Weights are calculated by special formulas, i.e. *term frequency method* (*tf*) or *term frequency – inversed document frequency method* (*tf – idf*) [18].

The *idf* method is described by the formula below:

$$idf_t = \log \frac{N}{df_t} \tag{11}$$

where:

N – the number of analysed documents,

df_t – the number of documents where term t occurs.

The *tf – idf* method is described by the formula below:

$$tf - idf_{t,d} = \log \frac{N}{df_t} \times tf_{t,d} \tag{12}$$

where:

$tf_{t,d}$ – the number of times that term t occurs in document d.

Table 10

Values of similarity of the sentences using popular methods. Describe of columns (experiments): Col. 1 – Similarity method based on Lda without coding terms method; Col. 2 – Cosine distance based on term frequency weight method (tf); Col. 3 – Dice distance based on tf weight method; Col. 4 – Jaccard distance based on tf weight method; No. – number of sentence. Experiments 2–4 based on Table 8 data (i.e. method of coding synonym terms was used)

No.	Col. 1	Col. 2	Col. 3	Col. 4
s1	0.80	0.40	0.08	0.25
s2	0.67	0.55	0.10	0.37
s3	0.75	0.25	0.06	0.14
s4	0.33	0.00	0.00	0.00
s5	0.57	0.46	0.07	0.30
s6	0.71	0.77	0.12	0.62
s7	0.75	0.63	0.07	0.45
s8	0.83	0.66	0.11	0.50
s9	0.50	0.25	0.62	0.14
s10	0.40	0.22	0.50	0.12
s11	0.62	0.40	0.53	0.25
s12	0.88	0.90	0.00	0.81
s13	0.78	0.56	0.06	0.38
s14	0.83	0.83	0.13	0.71
s15	0.71	0.69	0.04	0.52
s16	0.71	0.57	0.08	0.40
s17	0.88	0.81	0.05	0.68

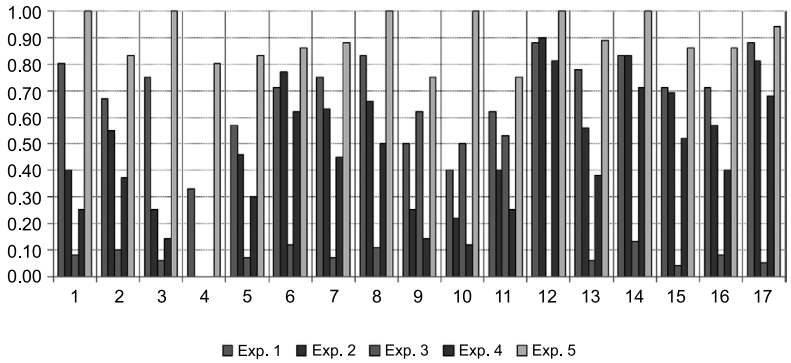

Fig. 3. Graphical results of similarity of the incorrect sentences using popular methods. Description of experiments: Exp. 1 – Similarity method based on Lda without coding terms method; Exp. 2 – Cosine distance based on term frequency weight method (*tf*); Exp. 3 – Dice distance based on *tf* weight method; Exp. 4 – Jaccard distance based on *tf* weight method; Exp. 5 – Similarity between correct and incorrect sentences (with using method of terms coding) based on Table 8 data

As can be seen, the described similarity methods (based on Lda) include a fully different algorithm compared with the popular methods (based on terms weights and vector space models) because of the identification of the distribution of terms. Described popular methods based on the number of occurrences of terms in documents only. Methods based on Lda do not need other documents instead of e.g. term frequency – inversed document frequency method to estimate weight of terms (in case of two sentences impossible to apply). Graphical interpretation includes all described methods is shown below (Figure 3). The best results includes exp. no. 5.

6. Summary

The method of coding terms described in this paper increases the precision of calculation of the similarity measures based on Levenshtein distance significantly. This method is characterized by the speed of data analysis and simplicity of implementation. The coding method of terms in combination with the Levenshtein distance and the similarity measures can be used in: detecting plagiarism (resignation of variety of nouns and verbs based on standard thesaurus and stemming algorithms), finding phrases in text documents [8] (or web documents [13], etc.), algorithms for correcting mistakes, mechanism of identification and classification of content based on term weighted methods [1, 14, 15], etc.

The proposed solutions are applied and have been tested in the mechanism of topic analysis and descriptions of selected written works (diploma thesis) to automatic selection of supervisors and reviewers at the Faculty of Physics, Mathematics and Computer Science of the Cracow University of Technology[2]. The solution also was included in Anti-plagiarism System of Faculty of Physics, Mathematics and Computer Science[3]. Tests results show a high quality of the text mining analysis.

References

[1] Niewiarowski A., *Term frequency optimization for the vector space model*, "Technical Transactions", 9-M/2012, 155-165.

[2] Yih W., Meek Ch., *Improving Similarity Measures for Short Segments of Text*, Microsoft Research, USA 2007.

[3] Long-Scheng Cz., Chia-Wei Ch., *A New Term Weighting Method by Introducing Class Information for Sentiment Classification of Textual Data*, "Proceedings of the International MultiConference of Engineers and Computer Scientists", IMECS 2011, 394-397.

[4] Metzler D., Dumais S., Meek Ch., *Similarity Measures for Short Segments of Text*, Microsoft Research, USA 2007.

[5] Piasecki M., Broda B., *Semantic similarity measure of Polish nouns based on linguistic features*, "Business Information Systems 10th International Conference, Lecture Notes in Computer Science, Springer", BIS 2007, 381-390.

[2] Manager of Diploma Thesis web site: https://dyplomy.fmi.pk.edu.pl

[3] Screenshot of our .NET application (for MS Windows) for analyze plagiarism: www.pk.edu.pl/~aniewiarowski/programy/antyplagius.png

[6] Novay L.G., Novay Ch. W., Brussee R., *Thesaurus Based Term Ranking for Keyword Extraction*, "DEXA'10 Proceedings of the 2010 Workshops on Database and Expert Systems Applications, Computer Society", 2010.
[7] Castillo Sequera J.L., Fernandez del Castillo Diez J.R., Gonzales Sotos L., *A clustering algorithm based on a recursive function of distance and similarity*, "IADIS European Conference Data Mining" 2011, 43-50.
[8] Szwed P., *Conecpts extraction from unstructered Polish texts: a rule based approach*, "Proceedings of the 2015 Federated Conference on Computer Science and Information Systems, Springer", 355-364.
[9] Lovins J.B., *Development of a Stemming Algorithm*, "Mechanical Translation and Computational Linguistics" vol. 11, nos. 1 and 2 1968.
[10] Willett P., *The Porter stemming algorithm: then and now*, "Program", Vol. 40, 219-223.
[11] Abramowicz W., Filipowska A., Małyszko J., Wagner T., *Lemmatization of Multi-Word Entity Names for Polish Language Using Rules Automatically Generated Based on the Corpus Analysis*, "Language and Technology Conference", 2009, 540-544.
[12] Niewiarowski A., Stanuszek M., *Parallelization of the Levenshtein distance algorithm*, "Technical Transactions", 3-NP/2014, 109-122.
[13] Niewiarowski A., *Działanie parsera Part-of-Speech Tagging w ujęciu mechanizmu Web Content Mining*, 6'th National Conference „Science and Industry", 2011, 93-100.
[14] Niewiarowski A., Stanuszek M., *The mechanism of identification and classification of content*, "Studia Informatica", Volume 34, Number 2B (112), 2013, 205-222.
[15] Niewiarowski A., *Mechanism of plagiarism detection based on the variation of the Levenshtein distance algorithm*, 5'th National Conference „Science and Industry", 2010, 86-103.
[16] Левенштейн В.И., *Двоичные коды с исправлением выпадений, вставок и замещений символов*, „Доклады Академий Наук СССР", 163 (4), 1965, 845-848.
[17] Singhal, Amit., *Modern Information Retrieval: A Brief Overview*, "Bulletin of the IEEE Computer Society Technical Committee on Data Engineering", 24 (4), 2001, 35-43.
[18] Rajaraman, A., Ullman, J.D., *Data Mining*, "Mining of Massive Datasets", Cambridge University Press, 2014, 1-17.

Appendix 1

Table 11

Example of Polish sentences. Stars define the same origin (i.e. method of coding synonyms was used)

No.	Correct sentence	Incorrect sentence
s1	Jutro będzie nowy* dzień	Jutro bedzie nowiutki* dzien
s2	Gdy** chcesz opisać prawdę, elegancję pozostaw*** krawcom.	Kiedy** chcesz opisac prawde, elegancje zostaw*** krawcom.
s3	Kto się lęka**** już przegrał	Kto sie boi**** juz przegral

Table 12

Values of similarity of the sentences using popular methods. Description of columns (experiments): Col. 1 – Similarity method based on Lda without coding terms method; Col. 2 – Similarity method based on Lda with coding terms method; Col. 3 – Cosine distance based on term frequency weight method (tf); Col. 4 – Dice distance based on tf; Col. 5 – Jaccard distance based on tf weight method; No. – number of sentence

No.	Col. 1	Col. 2	Col. 3	Col. 4	Col. 5
s1	0.75	1.00	0.25	0.06	0.14
s2	0.86	1.00	0.57	0.08	0.40
s3	0.40	0.60	0.40	0.08	0.25

Table 11 includes correct and incorrect Polish sentences. Sentences have similar meaning but include different terms. Column no. 2 in table 12 includes the best values of the tests.

Table 13

Example of Russian sentences. Stars define the same origin (i.e. method of coding synonyms was used)

No.	Correct sentence	Incorrect sentence
s1	Завтра будет новый* день	Завтра будетъ новенкий* день
s2	Если** вы хотите сказать*** правду, оставьте элегантность портным	Когда** вы хатите рассказать*** правду,оставте элегантность портным
s3	Кто боится уже проиграл	Кто баиться уже праиграл

Table 14

Values of similarity of the sentences using popular methods. Description of columns (experiments): Col. 1 – Similarity method based on Lda without coding terms method; Col. 2 – Similarity method based on Lda with coding terms method; Col. 3 – Cosine distance based on term frequency weight method (tf); Col. 4 – Dice distance based on tf; Col. 5 – Jaccard distance based on tf weight method; No. – number of sentence

No.	Col. 1	Col. 2	Col. 3	Col. 4	Col. 5
s1	0.75	1.00	0.75	0.18	0.60
s2	0.75	1.00	0.75	0.09	0.60
s3	0.75	0.75	0.50	0.12	0.33

Table 13 includes correct and incorrect Russian sentences. Sentences have similar meaning but include different terms. Column no. 2 in table 14 includes the best values of the tests.

Table 15

Example of Belarusian sentences. Stars define the same origin (i.e. method of coding synonyms was used)

No.	Correct sentence	Incorrect sentence
s1	Заўтра* будзе новы дзень	Заутра* будзіць новый дзень
s2	Калі вы хочаце сказаць праўду, пакіньце** элегантнасць для краўцоў	Калі вы хочаце сказать правду, пазастаўце** элегантнасць для краўцоў
s3	Хто баіцца ўжо прайграў	Кто баіца ужо праіграў

Table 16

Values of similarity of the sentences using popular methods. Description of columns (experiments): Col. 1 – Similarity method based on Lda without coding terms method; Col. 2 – Similarity method based on Lda with coding terms method; Col. 3 – Cosine distance based on term frequency weight method (tf); Col. 4 – Dice distance based on tf; Col. 5 – Jaccard distance based on tf weight method; No. – number of sentence

No.	Col. 1	Col. 2	Col. 3	Col. 4	Col. 5
s1	0.75	0.75	0.50	0.12	0.33
s2	0.89	1.00	0.77	0.08	0.63
s3	0.50	0.50	0	0	0

Table 15 includes correct and incorrect Belarusian sentences. Sentences have similar meaning but include different terms. Column no. 2 in table 16 includes the best values of the tests.