Computer Assisted Methods in Engineering and Science, **26**: 163–175, 2019, doi: 10.24423/cames.277

Similarity detection based on document matrix model and edit distance algorithm

Artur Niewiarowski

Cracow University of Technology
Department of Computer Science
Faculty of Computer Science and Telecommunications
Warszawska 24, 31-155 Cracow, Poland
e-mail: artur.niewiarowski@pk.edu.pl

This paper presents a new algorithm with an objective of analyzing the similarity measure between two text documents. Specifically, the main idea of the implemented method is based on the structure of the so-called "edit distance matrix" (similarity matrix). Elements of this matrix are filled with a formula based on Levenshtein distances between sequences of sentences. The Levenshtein distance algorithm (LDA) is used as a replacement for various implementations of stemming or lemmatization methods. Additionally, the proposed algorithm is fast, precise, and may be implemented for analyzing very large documents (e.g., books, diploma works, newspapers, etc.). Moreover, it seems to be versatile for the most common European languages such as Polish, English, German, French and Russian. The presented tool is intended for all employees and students of the university to detect the level of similarity regarding analyzed documents. Results obtained in the paper were confirmed in the tests shown in the article.

Keywords: plagiarism detection, plagiarism system, edit distance, Levenshtein distance, similarity measure, text mining, information retrieval.

1. Introduction

Algorithms for the analysis of similarity in text documents find multiple applications in various fields, especially in web search engines, grammar and spellchecker subprograms, and, most importantly – plagiarism detection systems. Specifically, plagiarism detection is the process of locating similar parts within the texts of two or more documents.

There are many various algorithms with different solutions implemented in the plagiarism detection mechanisms, e.g., the bag of words method [1], stylometry [11], string matching and others. Most algorithms are based on thesaurus [9] and lemmatization methods adapted to the specific languages [15, 17, 18]. In most instances, applied approaches slow down the whole mechanism. Thus, it is necessary to come up with new, universal and faster plagiarism detection algorithms.

This article describes a new concept of the plagiarism detection algorithm. The proposed method of plagiarism detection is characterized by the high speed of computations as well as it does not need an implementation of any additional thesauruses. The algorithm is developed for analyzing very large documents (e.g., books, diploma works, newspapers, etc.), and it is universal for most common European languages (e.g., Polish, English, German, French, Russian, etc.). The mechanism was implemented at the Cracow University of Technology as *Internal Plagiarism System – Antyplagius*. Test results are included in the article.

2. Description of the problem

The main problem is to find the universal algorithm for fast analyzing of plagiarism between two text documents written in different languages. Moreover, it is important to find such a mechanism without importing any dedicated thesauruses and implementing lemmatization and stemming processes [2, 3, 5, 7, 9, 12, 14, 16].

The proposed mechanism of the analysis of the similarity of two documents is mainly based on the edit distance (more precisely, the Levenshtein distance) and on the structure of a matrix representing two compared documents. The Levenshtein distance [6] between two strings of characters is equal to the minimum number of insertions, deletions and substitutions of characters required to convert one string into the second one. Levenshtein distance has applications in many areas, e.g., text analysis (detection of plagiarism), spell-checking in text processors [8], web mining (search engine robots) [13], bioinformatics (Levenshtein-Damerau distance for DNA structure analysis [4]), etc. In this case, the LDA is used instead of several implementations of stemming and lemmatization methods. This approach is possible when the described below matrix of documents is applied. Furthermore, this matrix allows showing graphical results for users. The mechanism of plagiarism detection is universal for most European languages.

2.1. Description of the Levenshtein distance algorithm

The Levenshtein distance k for two strings is a minimum number of operations: insertion, deletion and substitution required to convert one term (string) into the other. The Levenshtein distance k is equal to the $\mathbf{D}[m,n]$ element of the so-called Levenshtein matrix $\mathbf{D}$:

$$k = \mathbf{D}[m,n] = \text{LevenshteinDistance (Text1, Text2)}.$$

The main idea of the LDA (Levenshtein Distance function) is described by the following formula:

$$\begin{cases} \prod_{i=1}^{n}\prod_{j=1}^{m} \mathbf{D}[i,j] = \min(d[i-1,j]+1, d[i,j-1]+1, d[i-1,j-1]+cost), \\ cost = 0 : \mathbf{Text1}[i] \equiv \mathbf{Text2}[j], \\ cost = 1 : \mathbf{Text1}[i] \neq \mathbf{Text2}[j], \\ \mathbf{D}[i,0] = i, \\ \mathbf{D}[0,j] = j, \\ \mathbf{D}[0,0] = 0, \end{cases} \tag{1}$$

where $\mathbf{D}$ – Levenshtein matrix of the size $n+1$, $m+1$, formed for two text strings: *Text1* and *Text2*, m, n – lengths of two strings respectively, $\mathbf{D}[i,j]-(i,j)$ – element of Levenshtein matrix $\mathbf{D}$, *min* – a function to calculate minimum of three variables, *cost* – a variable that gets values either 0 or 1.

The LDA is described by the following pseudo-code:

```
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] :=
                    min(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];
```

□

The figure below and the pseudo-codes above show that value of element $[i,j]$ of matrix **D** in a current iteration is calculated based on the values: $D[i-1,j]$, $D[i,j-1]$ and $D[i-1,\ j-1]$ for the Levenshtein algorithm. It means that each of these values must be calculated in the previous iterations of the algorithm.

		y	e	s	t	e	r	d	a	y
	0	1	2	3	4	5	6	7	8	9
t	1	**1**	2	3	3	4	5	6	7	8
o	2	2	2	3	4	4	5	6	7	8
m	3	3	3	3	4	5	5	6	7	8
o	4	4	4	4	4	5	6	6	7	8
r	5	5	5	5	5	5	5	6	7	8
r	6	6	6	6	6	**6**	5	6	7	8
o	7	7	7	7	7	7	6	6	7	8
w	8	8	8	8	8	8	7	7	7	**8**

Fig. 1. Levenshtein matrix D, constructed for terms: yesterday and tomorrow[1].

Some practical examples of the calculation of the Levenshtein distance for other more complex text strings are presented in Table 1.

Table 1. Examples of the Levenshtein distance between two strings.

No.	String no. 1	String no. 2	Levenshtein distance
1	Dog	Dogs	1
2	University	Universities	3
3	Tom is writing a letter	Tom is writin letters	4

In the first example, we need to add one character in string no. 1 or remove one character in string no. 2 to transform one string into the other. In the second case, we need to substitute one character and add two characters (string no. 1) or substitute one character and remove two characters (string no. 2). In the last example, we need to remove four characters ("g", "a", "s" and space) in string no. 1 or add four characters in string no. 2. All of this explains the measure of Levenshtein distance calculated in column four.

[1] The application for calculations of both Levenshtein and Levenshtein-Damerau matrices is available from the web site: www.pk.edu.pl/~aniewiarowski/publ/levenMatrix.exe.

2.2. Implementation of Levenshtein distance to calculate a similarity measure

The LDA can be used for analyzing the difference between two terms expressed as a value between $\langle 0,\ 1\rangle$ (i.e., a measure of similarity) [10].

The measure of similarity p is the quotient of the number of Levenshtein operations (after calculation of the LDA) by the number of all Levenshtein operations in the pessimistic case in which two strings are completely different.

The similarity measure p is calculated by the formula:

$$p = 1 - \left(\frac{k}{k_{\max}}\right); \qquad k_{\max} = \max(n, m), \qquad \begin{array}{c} k \geq 0, m > 0, n > 0, \\ p \in \langle 0, 1\rangle, \end{array} \tag{2}$$

where $k_{\max}$ – length of the longest of analyzed two terms/text strings (i.e., pessimistic case where k is equal to the length of the longest term).

Based on (2), some examples for the calculation of the similarity measure for simple and complex sentences are presented in Table 2.

Table 2. Examples of the similarity measure based on Levenshtein distance between two short texts.

No.	Text no. 1	Text no. 2	k	$k_{\max}$	p
1	Dog	Dog**s**	1	4	0.75
2	University	Universitie**r**	3	12	0.75
3	Tom is writing a letter	Tom is **writin** letter**r**	4	23	0.82
4	World	World	0	5	1
5	Dog	Cat	3	3	0
6	XXXX	XX**YY**	2	4	0.5
7	Pięćdziesięciopięciogroszówka	Pi**ec**dziesi**e**ciopi**e**ciogrosz**o**wka	5	29	0.83
8	Компьютер	компьютер**ы**	1	11	0.90

The examples above are very interesting because most European languages have similar basic grammatical rules for inflected forms, and after changes, the terms look similar to their original forms. As can be seen, the LDA can also be used successfully for dealing with misspellings in the analyzed texts.

2.3. The concept of the matrix representation of two compared documents

In cases where phrases in two texts have different root forms (different meaning) or term in a sentence is replaced by another phrase (with the same meaning), the similarity matrix concept allows for a correct analysis of the similarity of texts. Such a case is presented below, showing the main idea of using the matrix distance model in similarity detection.

Figure 2 shows the graphical interpretation of the similarity between two identical documents (texts). Horizontal and vertical axes represent positions of terms (words) in the documents. Punctuation marks such as commas, dots, question marks, etc. were automatically removed from the texts for the analysis. Each point in Fig. 2 stands for the occurrence (the original position) of the term from the document no. 1 in the document no. 2.

As we can see, some of the terms are repeated many times in the documents, but only the terms located diagonally reveal some information about the identity. This means that the diagonal distribution of the sequence for terms, even those moved parallelly, shows the similarity of the text.

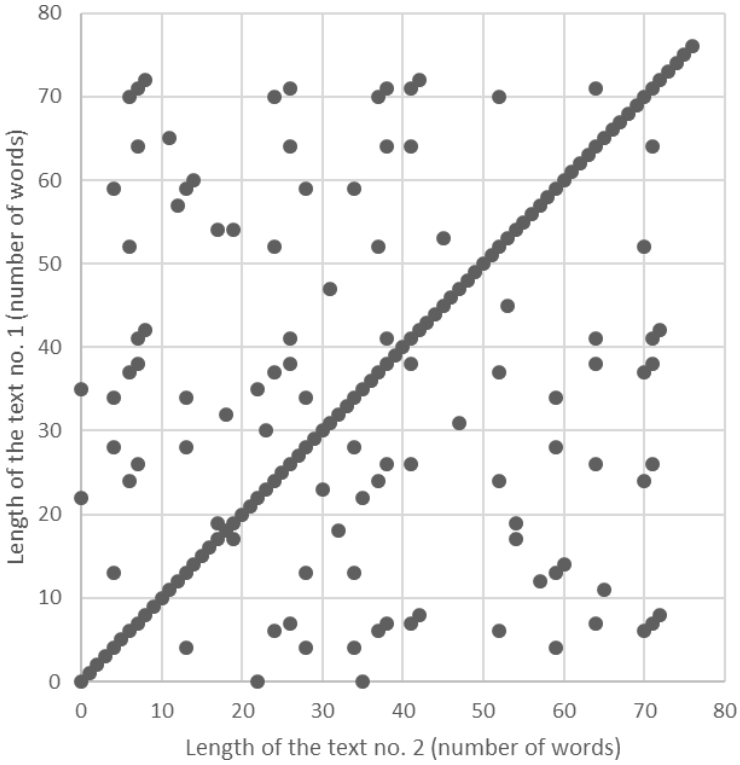

Fig. 2. The graphical visualization of the similarity between two identical documents (texts)[2] based on the matrix structure concept. Result of comparison: 100%.

The main idea of composing of the matrix for two documents is described by the following formula:

$$\begin{cases} \underset{ti=1}{\overset{tm}{\mathrm{I}}} \underset{tj=1}{\overset{tn}{\mathrm{I}}} \mathbf{D}[ti,tj] = \beta, \\ \beta = \text{true} : \mathbf{Doc1}[ti] \equiv \mathbf{Doc2}[tj], \\ \beta = \text{false} : \mathbf{Doc1}[ti] \neq \mathbf{Doc2}[tj], \end{cases} \tag{3}$$

where $\mathbf{D}$ – documents matrix of the size tn, tm, created for documents: **Doc1** and **Doc2**, tm, tn – number of elements of two documents respectively, β – values either *false* or *true* (3), $\mathbf{D}[ti, tj]$ – (ti, tj) – element of documents matrix $\mathbf{D}$, $\mathbf{Doc1}[ti]$ – element (term) of document 1 separated by a space from the next element of the document.

The pseudo-code below describes the algorithm for composing the matrix representation $\mathbf{D}$ of two compared documents.

```
input variables: term Doc1[0..tm-1], term Doc2[0..tn-1]
  declare: boolean D[0..tm-1, 0..tn-1]
  for ti from 0 to tm-1
    for tj from 0 to tn-1
```

[2] Content of the documents written in Portuguese: *Mas, apesar da variedade de possibilidades que a voz possui, seria um instrumento de comunicação muito pobre se não se contasse com mais do que ela. A capacidade de expressão do homem não disporia de mais meios que a dos animais. A voz, sozinha, é para o homem apenas uma matéria informe, que para se converter num instrumento perfeito de comunicação deve ser submetida a um certo tratamento. Essa manipulação que a voz recebe são as "articulações".*

```
      if term of Doc1 at (ti) = term of Doc2 at (tj) then
        D[ti,tj] := true
      else
        D[ti,tj] := false
      end if

    end for (variable tj)
  end for (variable ti)
return m;
```

□

The example above describes the similarity between two documents based on the occurrences of the same terms. However, in real analysis, most of the documents are composed of the different inflected forms of the terms. Furthermore, in the sequence, some words may not be present or new terms may appear.

2.4. Implementation of the similarity measure p in the matrix representation of compared documents

The LDA may be used instead of lemmatization and stemming methods. Furthermore, it can be used successfully for getting around misspellings in texts.

The main idea of composing of matrix $\mathbf{M}$ with the measure of similarity p (formula (2)) based on edit distance is described by the following formula:

$$\begin{aligned} &\prod_{ti=1}^{tm}\prod_{tj=1}^{tn}\mathbf{M}[ti,tj]=\beta,\\ &\begin{cases}\beta=\text{true}: fp(\mathbf{Doc1}[ti],\mathbf{Doc2}[tj])\geq bp,\\ \beta=\text{false}: fp(\mathbf{Doc1}[ti],\mathbf{Doc2}[tj])<bp,\end{cases}\end{aligned} \tag{4}$$

where fp – function returns similarity measure p (formula (2)), bp – acceptable boundary value of similarity measure parameterized by the user (e.g., corresponds to the type of the documents).

The pseudo-code below describes an algorithm of composing the documents matrix $\mathbf{M}$ with the measure of similarity p (2) calculated based on the edit distance.

```
input variables: term Doc1[0..tm-1], term Doc2[0..tn-1],
                 decimal bp
  declare: boolean M[0..tm-1, 0..tn-1]
  for ti from 0 to tm-1
    for tj from 0 to tn-1

    if fp(term of Doc1 at (ti), term of Doc2 at (tj)) >= bp then
        M[ti,tj] := true
      else
        M[ti,tj] := false
      end if

    end for (variable tj)
  end for (variable ti)
return M;
```

□

The graphical visualization of the similarity between two identical in meaning but written in Portuguese and Spanish texts are shown in Figs 3 and 4. This clearly demonstrates the usage of the edit distance algorithm.

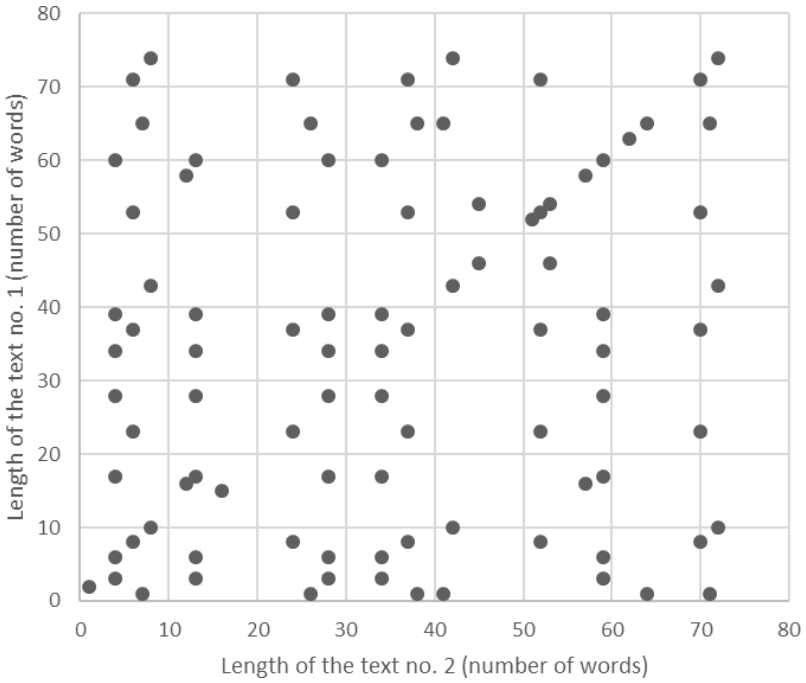

Fig. 3. The graphical visualization of the similarity between two short documents (texts)[3] written in two similar languages: Portuguese and Spanish, without using a Levenshtein distance.

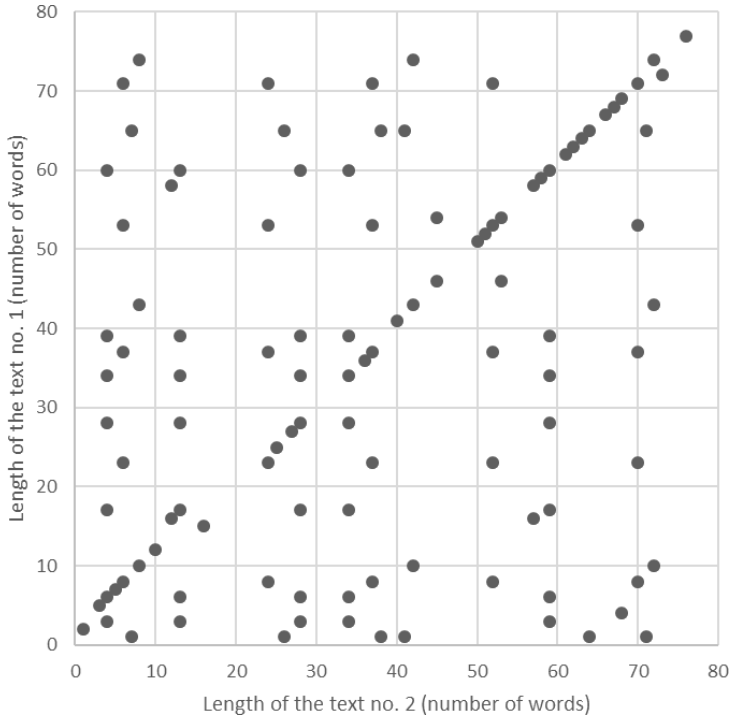

Fig. 4. The graphical visualization of the similarity between two short documents (texts)[3] written in two similar languages: Portuguese and Spanish, with a Levenshtein distance included.

[3] Content of the documents written in Spanish: *Pero, a pesar de esta variedad de posibilidades que la voz posee, sería un muy pobre instrumento de comunicación si no contara más que con ella. La capacidad de expresión del hombre no dispondría de más medios que la de los animales. La voz, sola, es para el hombre escasamente una materia informe, que para convertirse en un instrumento perfecto de comunicación debe ser sometida a un cierto tratamiento. Esa manipulación que recibe la voz son las "articulaciones".*

Comparing the results of proposed mechanisms without using a Levenshtein distance (Fig. 3, result of comparison: 30.77%) and with a Levenshtein distance included (Fig. 4, result of comparison: 53.85%, $bp >= 0.75$) shows some additional new points located in the sequence on the diagonal of the diagram. The points which do not create the sequence can be treated as some kind of 'noise', and they can be removed by an additional filter algorithm (Fig. 5, which is not the subject of this work). Edit distance can be used instead of stemming and lemmatization methods because terms that have a different meaning with the high similarity at the same time do not create the sequence. As can be seen in Fig. 4, the Levenshtein distance increases the accuracy of the algorithm significantly.

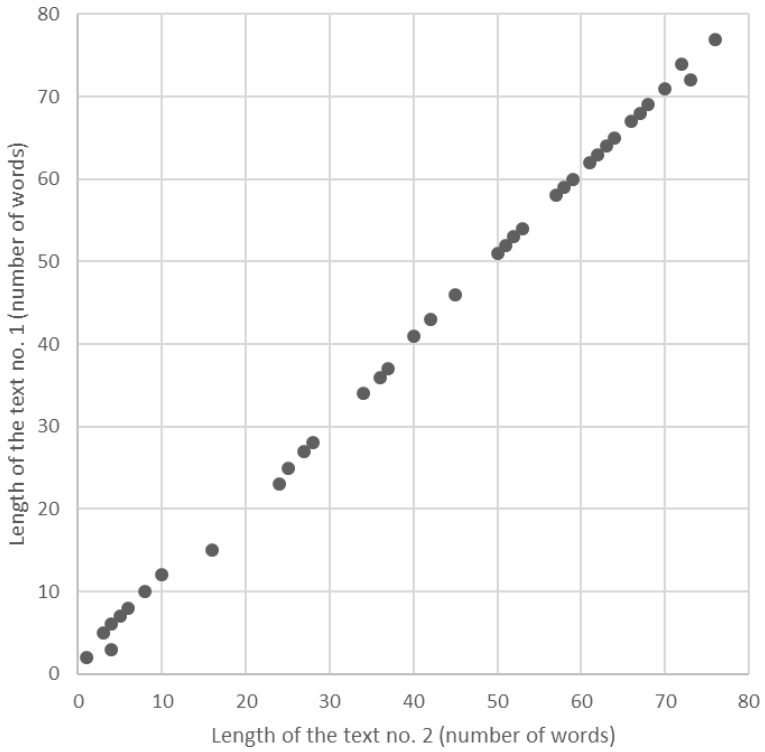

Fig. 5. The graphical visualization of the similarity between two short documents after applying the filter removing incorrect points.

3. Numerical implementation

The tests of the presented algorithm were performed by using the *Internal Plagiarism System – Antyplagius* installed at the Cracow University of Technology.

3.1. Analysis of two large fragments of books

Firstly, the Vampire Chronicles (Anne Rice, approx. 9000 words) and Robinson Crusoe (Daniel Defoe, approx. 13 000 words) were analyzed with the similarity p (2) between words assumed bigger than 0.75. The results of the analysis are shown in Figs 6 and 7.

As can be seen in Fig. 6, the chart without zooming does not give any information about the similarity of the texts. It may even seem that the texts are similar. However, the 'zoomed in' visualization shows that the points do not form any linear sequence (Fig. 7).

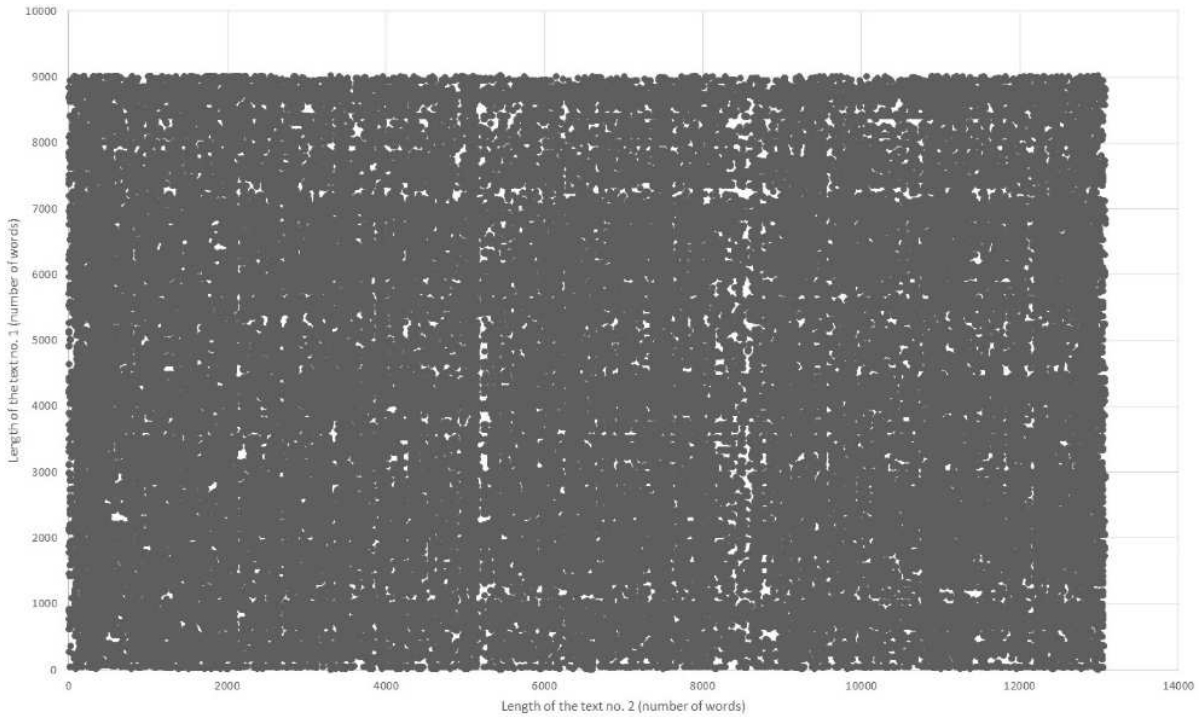

Fig. 6. The graphical visualization of the similarity between two books: Vampire Chronicles (Anne Rice) and Robinson Crusoe (Daniel Defoe).

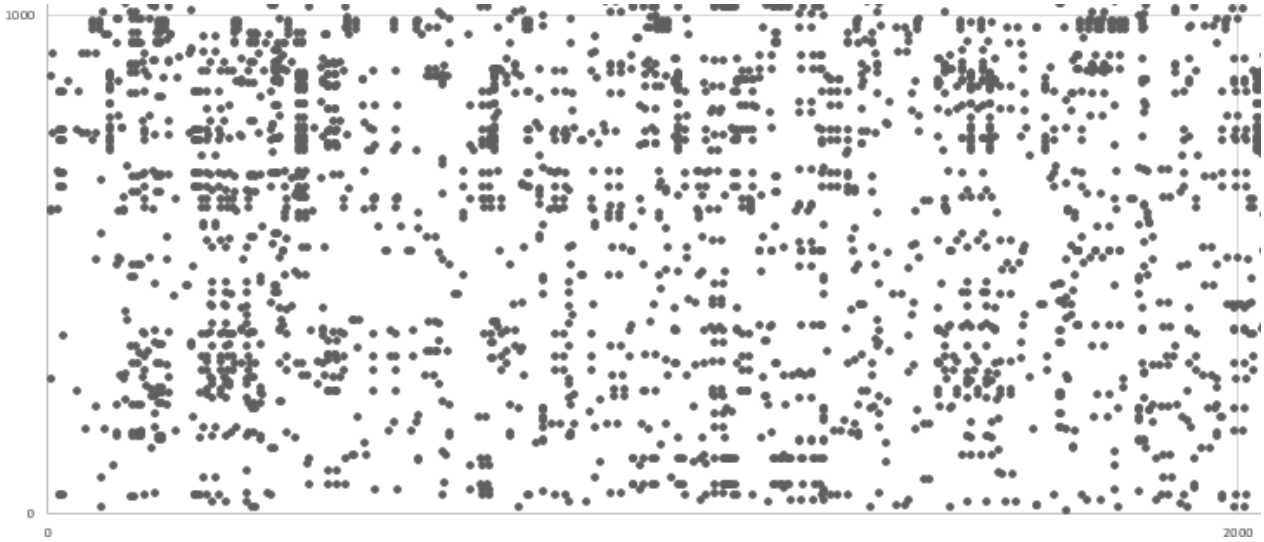

Fig. 7. Fragment of the graphical visualization of the similarity between two books: Vampire Chronicles (Anne Rice) and Robinson Crusoe (Daniel Defoe).

3.2. Analysis of two full fragments of the same book

Two full fragments of the book 'Robinson Crusoe' (Daniel Defoe, approx. 13 000 words) were analyzed with the similarity p (2) between the words assumed bigger than 0.75, with no filtering and skipping words shorter than 8 characters. The results of the analysis are shown in Figs 8 and 9.

Figures 8 and 9 show the obtained similarity graphs based on the analysis of the same book. Figure 9 shows clearly the domains where points create the linear sequences (diagonal lines). These places are fragments of similarity between the compared texts.

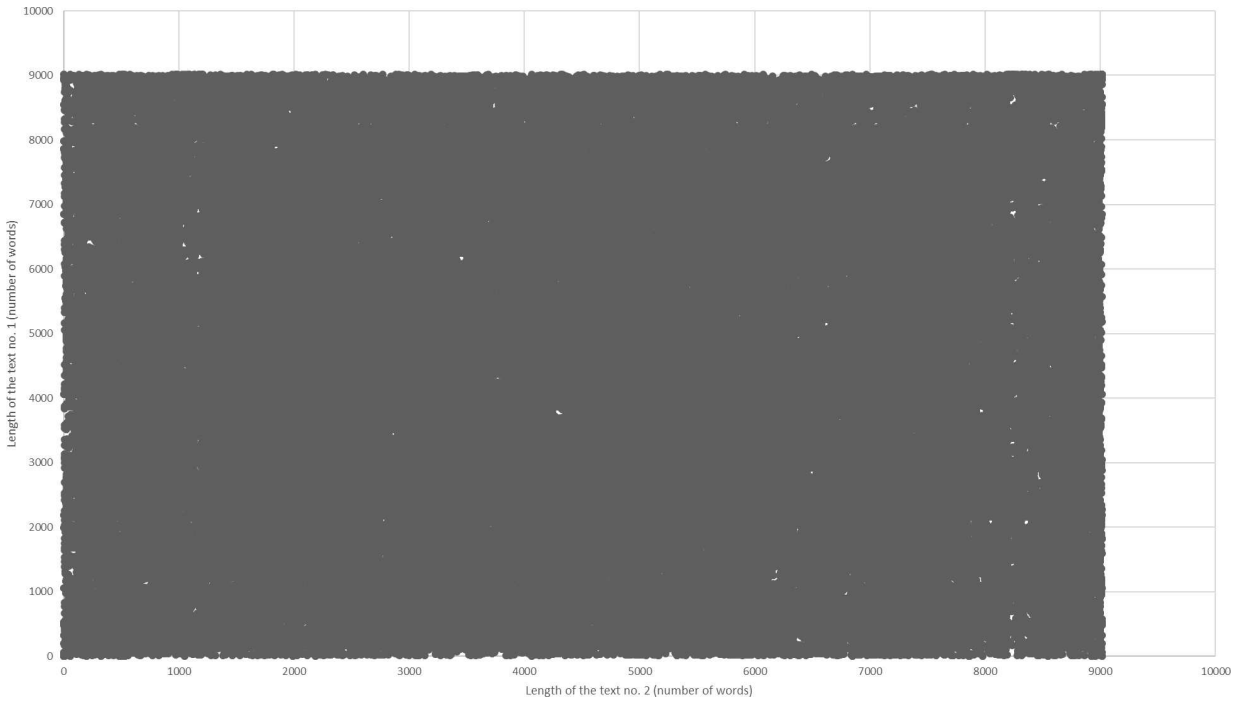

Fig. 8. The graphical visualization of the similarity between full fragments of the same book – Robinson Crusoe (Daniel Defoe).

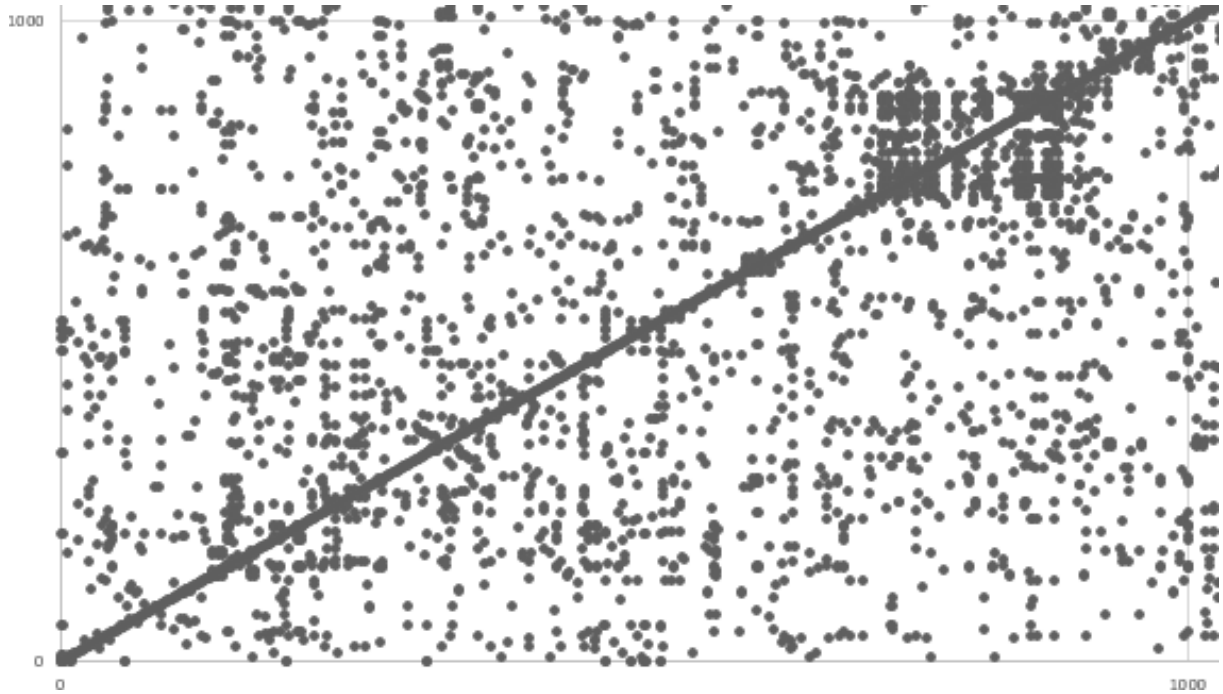

Fig. 9. Fragment of the graphical visualization of the similarity between full fragments of the Robinson Crusoe (Daniel Defoe).

3.3. Analysis of two documents about C# .NET programming language

A homework of a foreign student written in Polish and a teacher's database of texts (translated from Wikipedia and other sources by Google Translate – from Russian, Ukrainian and Belarusian into Polish) were analyzed.

Original texts are included in the Appendices 1 and 2. As can be seen in Fig. 10, points create the sequence, which means that spelling and grammatical errors did not affect the correct finding of similar fragments in the analyzed texts. Parameters for the calculations: similarity between words: 0.75, filtering: no skipping words, skipping numbers.

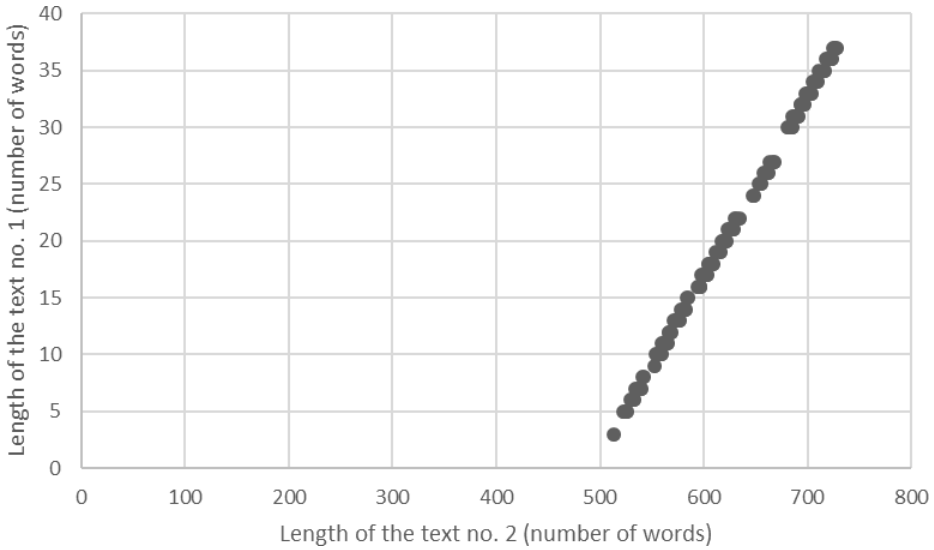

Fig. 10. Text no. 1 represents a fragment of the teacher database (Appendix 1), text no. 2 represents homework (Appendix 2).

4. Conclusions and future work

Tests document that the method proposed in the present contribution constitutes a significant improvement of the text analysis and is competitive with other methods that serve the same objective. Furthermore, the method does not require thesauruses, lemmatization and stemming methods. Additionally, it is versatile for most European languages. Further work will concern the use of the above tests in stylometry [11].

Appendix 1

[...] mocna platforma dla stworzenia aplikacji Mogęr odznaczyć jej najważniejsze cechyObsługa wielu języków Podstawą platformy jest ogólnejęzykowe środowisko wzkonawcze Common Language Runtime CLR pozwalając NET obsługować wiele języków łącznie z C # VBNET C ++ F # a również różne dialekty innych języków przywiązane do NET na przykład Delphi NET Podczas kompilowania kodu w każdym z tych języków jest kompilowany dwzespole w CIL języka wspólnego Common Intermediate Language rodzaj platformy NET W związku z tym możemy zrobić wielu poszczególnych modułów aplikacji w poszczególnych językachCrossplatform NET Jest przenośną platformą z pewnymi ograniczeniami Na przykład najnowsza wersja platformy w chwili NET Framework obsługuje w większości nowoczesne systemy operacyjne Windows Windows Vista A dzięki projektu Mono można tworzyć aplikacje które działają na [...] rodziny LINUX w tam liku i na mobilnach platformach Android lub iOSPotężna biblioteka klasów NET – to jedyna dla wszystkich języków obsługiwanych biblioteka klasów I bez względu na to Jaką aplikację zbieramy się pisać w C# edytor tekstu czat lub webstrona – my będziemy wykorzystali bibliotekę klasów NETRóżnorodność technologii Wspólny język wykonawczy CLR i biblioteka klas bazowej są podstawami dla całego steku technologii które programiści mogą wykorzystać do budowy różnych

aplikacji Na przykład do pracy z bazami danych w tzm stoku danych mamy ADONET technology Do tworzenia aplikacji graficznej z bogatym interfejsem technologii WPF Do tworzenia [...]

Appendix 2

[...] Można wybrać następujące jego główne cechy są Wsparcie dla wielu języków Podstawą platformy jest Common Language Runtime Common Language Runtime CLR pozwalając NET obsługuje wiele języków łącznie z C # jest jak VBNET C ++ F # a różnymi dialektami innych językach przywiązane do NET na przykład Delphi NET Podczas kompilowania kodu w każdym z tych języków jest kompilowany do zespołu w CIL języka wspólnego Common Intermediate Language rodzaj platformy NET W związku z tym możemy poszczególnych modułów aplikacji w poszczególnych językach Crossplatform NET Jest przenośną platformę z pewnymi ograniczeniami Na przykład najnowsza wersja platformy w chwili NET Framework Obsługiwane w większości systemów operacyjnych Windows nowoczesny Windows Vista A dzięki projektu Mono można tworzyć aplikacje które działają na [...] rodziny OS Linux w tym platform mobilnych Android i iOS Potężny biblioteki klas NET Jest taka sama dla wszystkich języków obsługiwanych lekcje biblioteczne I bez względu na to co aplikacja nie będziemy pisać w C # edytor tekstu czat lub złożone strona tak czy będzie kolejna używamy NET biblioteki klasy Różnorodność technologii Wspólny język wykonawczego CLR i biblioteki klasy bazowej są podstawą dla całego stosu technologii które programiści mogą wykorzystać do budowy różnych aplikacji Na przykład do pracy z bazami danych w tej technologii stos jest ADONET technology Do tworzenia aplikacji graficznych z bogatym interfejsem technologii WPF Do tworzenia [...]

References

[1] S. Albitar, S. Fournier, B. Espinasse. An effective TF/IDF-based text-to-text semantic similarity measure for text classification. In: *Proceedings of the Web information systems engineering – WISE 2014. Part I*, pp. 105–114, 15th International Conference, Thessaloniki, Greece, October 12–14, 2014.

[2] J. Dawson. Suffix removal and word conflation. *ALLC Bulletin*, **2**(3): 33–46, 1974.

[3] A. Dziob, M. Piasecki. Implementation of the verb model in plWordNet 4.0. In: *Proceedings of the 9th Global Wordnet Conference*, pp. 114–123, Singapore, 8–12 January 2018.

[4] R. Gabrys, E. Yaakobi, O. Milenkovic. *Codes in the Damerau distance for DNA storage*. In: *2016 IEEE International Symposium on Information Theory (ISIT)*, Barcelona, Spain, pp. 2644–2648, 2016, doi: 10.1109/ISIT.2016.7541778.

[5] R. Krovetz. Viewing morphology as an inference process. In: *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, Pittsburgh, USA, pp. 191–202, 1993.

[6] V.I. Levenshtein. Binary codes for correcting dropouts, inserts, and symbol substitutions [in Russian: Двоичные коды с исправлением выпадений, вставок и замещений символов]. Reports of the Academy of Sciences of the USSR, **163**(4): 845–848, 1965.

[7] J. Lovins. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, **11**(1–2): 22–31, 1968.

[8] M.M. Maulana, R. Arifudin, A. Alamsyah. Autocomplete and spell checking Levenshtein distance algorithm to getting Text Suggest Error Data Searching in Library. *Scientific Journal of Informatics*, **5**(1): 67–75, 2018.

[9] M. Maziarz, M. Piasecki, E. Rudnicka. plWordnet – the proces of making a thesaurus [in Polish: Słowosieć – polski wordnet. Proces tworzenia tezaurusa]. *Polonica*, **34**: 79–98, 2014.

[10] A. Niewiarowski. Short text similarity algorithm based on the edit distance and thesaurus [in Polish: Algorytm podobieństwa krótkich fragmentów tekstów oparty na odległości edycyjnej i słowniku wyrazów bliskoznacznych]. *Czasopismo Techniczne. Nauki Podstawowe/Technical Transactions/Fundamental Sciences*. **1**: 159–173, 2016.

[11] M. Piasecki, T. Walkowiak, M. Eder. Open stylometric system WebSty: integrated language processing, analysis and visualisation. *Computational Methods in Science and Technology*, **24**(1): 43–58, 2018.

[12] M.F. Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, **14**(3): 130–137, 1980.

[13] T.A. Runkler, J.C. Bezdek. Web mining with relational clustering. *International Journal of Approximate Reasoning*, **32**(2–3): 217–236, 2003.

[14] R.S. Sandeep, Ch. Vinay, M.S. Hemant. Strength and accuracy analysis of affix removal stemming algorithms. *International Journal of Computer Science and Information Technologies*, **4**(2): 265–269, 2013.

[15] V. Sandulescu, M. Ester. Detecting Singleton review spammers using semantic similarity. In: *Proceeding WWW'15 Companion Proceedings of the 24th International Conference on World Wide Web*, pp. 971–976, May 18–22, 2015, Florence, Italy.

[16] B. Wahiba, K. Abdessalem. A new stemmer to improve information retrieval. *International Journal of Network Security & Its Applications*, **5**(4): 143–154, 2013.

[17] D. Weiss. A survey of freely available Polish stemmers and evaluation of their applicability in information retrieval. In: *2nd Language and Technology Conference*, pp. 216–221, Poznań, Poland, 2005.

[18] D. Weiss. *Stempelator: A hybrid stemmer for the Polish language*. Research Report RA-002/05. Institute of Computing Science, Poznań University of Technology, Poland, 2005.