# Detecting Text Similarity Based on Discrete Wavelet Transformation

**Trung Hung Vo[1], Imre Felde[2], Phan Hieu Ho[3], Ngoc Anh Thi Nguyen[4]**

[1] The University of Danang – University of Technology and Education, 48 Cao Thang, Hai Chau, Da Nang, Vietnam, vthung@ute.udn.vn

[2] Óbudai Egyetem, H-1034 Budapest, Bécsi út 96/B, Budapest, Hungary, felde.imre@uni-obuda.hu

[3] The University of Danang, 41 Le Duan, Hai Chau, Da Nang, Vietnam, hophanhieu@ac.udn.vn

[4] The University of Danang – Unversity of Science and Education, 459 Ton Duc Thang, Da Nang, Vietnam, ngocanhnt@ued.udn.vn

*Abstract: Detecting similarities between texts is an important stage of many different applications such as text classification, plagiarism detection, fake news identification,... In this paper, we propose a new approach to detect similarities between texts based on the Discrete Wavelet Transform (DWT) method. Specifically, the available source documents are converted into a set of real numbers called DNAs (Deoxyribose Nucleic Acid) through DWT. To check the similarity of any text, we also use DWT to generate DNAs for that text and calculate the smallest Euclidean distance from these DNAs to the source DNAs. Finally, by comparing with a threshold, the distance values will indicate whether the evaluation text is similar to a certain source text or not. Experimental results demonstrate that our proposed algorithm is highly effective in detecting text similarity by testing on a standard data set at the Annual International Conference on Plagiarism Detection (Plagiarism Analysis, Authorship Identification, and Near-Duplicate detection – PAN).*

*Keywords: Text Similarity; Text analysis; Discrete Wavelet Transformation; Natural Language Processing; Fake New Detection*

# 1 Introduction

With a large amount of digital documents published publicly on the Internet every day, detecting copying based on text content is essential. According to survey data from Josephson Institute Center for Youth Ethics *(https://josephsoninstitute.org)* on

43,000 high school students, the results of admitting to cheating on tests through copying from other documents are as follows:

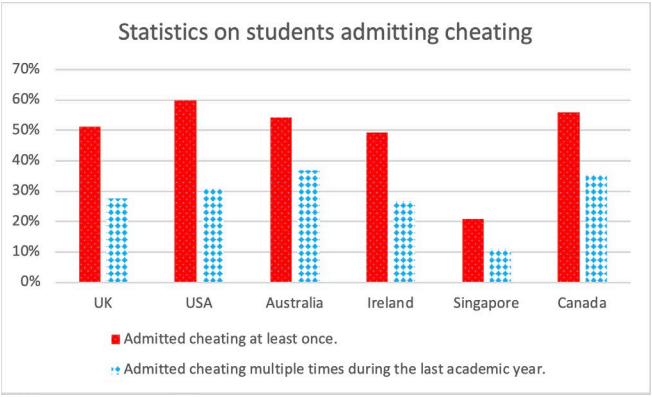


Figure 1

Statistics on students admitting cheating (Source: https://fixgerald.com)

There have been many studies on detecting text similarity to serve applications such as searching, detecting plagiarism, summarizing text, text mining, etc. Some typical results and many applications usefulness of measuring text similarity, including copy detection [1-4]. However, this problem still has many challenges related to data warehouse, techniques, algorithms, accuracy/reliability, efficiency, etc. Therefore, there are still many issues that need to be researched to find new approaches and solutions.

Detecting similar document content is a difficult problem that is of interest to many research groups. If the documents are exactly the same, it is easy to detect. However, most documents are copied and rewritten very sophisticatedly, making the problem much more difficult and the types of similarities are extremely diverse. A document can be copied in whole or only in part, the copied text parts can be changed such as added, edited, deleted or their positions are disturbed and located in any position of the new document. The new document after copying may only differ from the old text in a few small parts or may not be similar at all. In addition, there are also types of copying ideas, transliteration, etc. Because of the variability in text copying, no algorithm or technique can accurately measure the similarity between documents. Although this problem is not new, research is still needed to improve quality and speed.

Discrete Wavelet Transform (DWT) is mainly used in digital signal processing such as speech coding, image processing, applications in noise filtering, recognition and in other computer technology fields such as industry, telecommunications, electronics, medicine,... [10]. However, there is almost no research on DWT applied in the field of text processing.

In this study, we propose a completely new approach to detecting text similarity, which is based on the DWT method. This method is proposed and implemented through the main steps including: (1) The available original documents are converted into a set of real number sequences called source DNA through DWT; (2) To check the similarity of any text, we also use DWT to generate DNAs for that text and calculate the smallest Euclidean distance from these DNAs to the source DNAs; (3) Compared with a threshold, the distance values will indicate whether the evaluation document is similar to a certain source documents or not. Experimental results demonstrate that our proposed algorithm is highly effective in detecting text similarity by testing on a standard training data set of PAN and achieving an accuracy of over 97%.

The paper is organized into five main parts. Part two presents some research results related to the field of text similarity. The third part describes an overview of the proposed system and how to find DNA using the Haar filter. In the next section, we will analyze the proposed algorithm regarding data preprocessing, building a DNA set for the original text, and describing the similarity detection algorithm. The fifth part presents experimental results on the PAN standard data set and tests the proposed algorithm. Finally, we will give conclusions and future directions for the paper.

# 2 Related Research

Similarity between two documents is the similarity in content between those two documents (other factors such as images, special symbols, etc. are not considered). Therefore, two documents that are copies or nearly identical will have much similar content, or the "similarity" between the two documents will be high. Similarity lies in the range between 0 and 1. If the similarity is closer to 1, the possibility that the documents are similar is high and vice versa [5]. Therefore, to check whether the documents are similar or not, we must calculate the similarity between them. There are many methods to calculate text similarity such as based on string matching using Brute-Force, Morris-Pratt, Knuth-Morris-Pratt (KMP), Boyer-Moore, Karp-Rabin, Horspool, … [6]; Use vector space model to represent text and use distance measures to calculate similarity such as Euclid, Cosine, Jaccard, Dice,... [7]; vocabulary-based, database-based, and knowledge-based methods [8]; research on natural language processing... Each algorithm and matching method has a different approach and each algorithm has its own advantages and limitations.

One of the first tasks in text processing is to choose an appropriate text representation model to bring efficiency in calculation and processing. A text in raw form (character string) needs to be converted to another model to facilitate representation and calculation. Depending on each different processing algorithm, a separate representation model is chosen. In text processing problems, the vector

space model is most commonly used. This model represents text as a feature vector of terms/words appearing in the entire text set. Weighted features are often calculated through the TF-IDF measure [9].

Through the survey, we found that most of the methods used to detect copying are often based on a vector space model to represent text and use distance measures between vectors to calculate the similarity level. together. In general, research focuses on word matching using n-gram methods, fingerprints, and frequency statistics.

# 3 Proposal

## 3.1 General Model

With the goal of building a copy detection system, this study proposes an overview model of the system based on DWT [10], Haar filters [11] as shown in Figure 1.

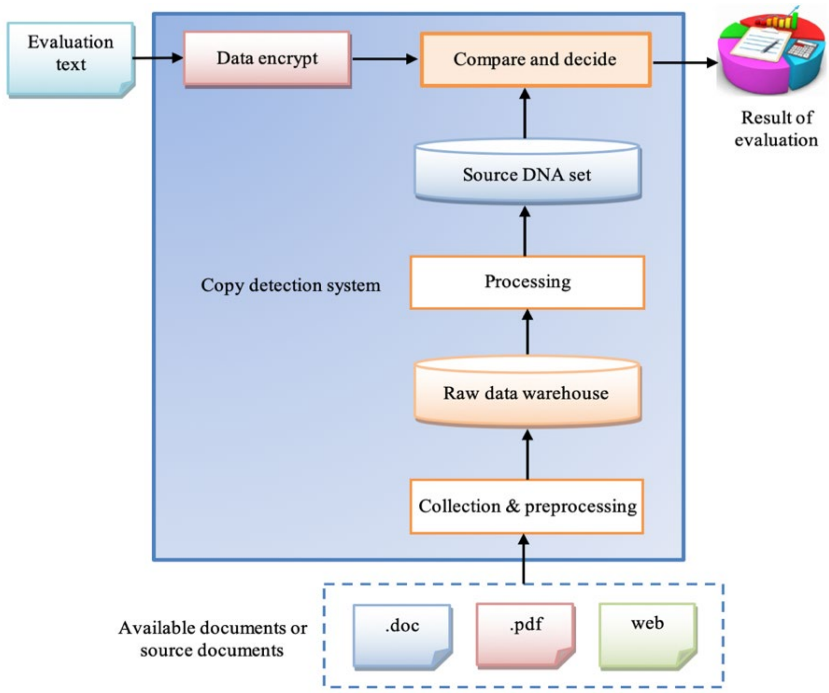


Figure 2
General model of copy detection system

This research focuses on designing blocks for a text copy detection system. First, available documents are collected, and preprocessing removes punctuation and special characters and stores them as raw data. In the preprocessing stage, the collected text will be segmented and sampled so that the samples have equal length. Then, these segments are stored as raw data for the purpose of extracting similar text segments (if any) at the evaluation result outputThe main idea here is that the documents will be digitized and passed through the Haar filter to get data for the source DNA set. Meanwhile, the review text is passed through the encoder for processing. This encoder is essentially the same as the DNA calculation process for the source text. However, the raw evaluation text produced after preprocessing will be segmented. Each paragraph in the evaluation text is then encoded into DNA. It aims to detect the similarity (if any) of that segment with another segment in the source data set. Therefore, raw evaluation text can be viewed as signal sequences that need to be processed.

Figure 2 describes in detail the processing process to evaluate the evaluation text against the source text set (data warehouse).

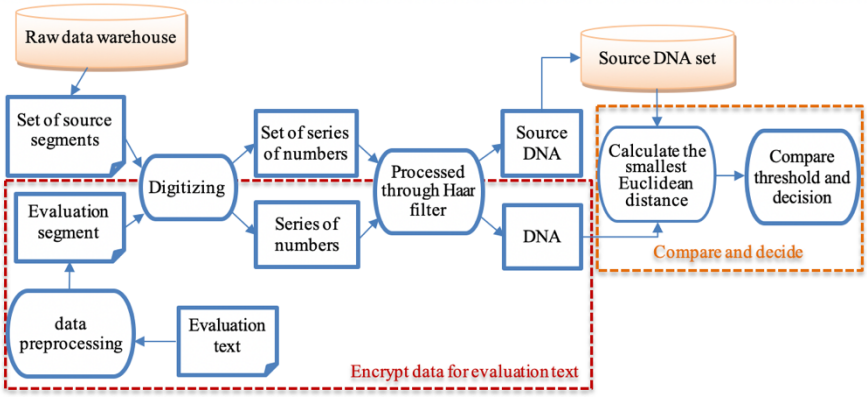


Figure 2

The diagram details the entire processing process to evaluate the test text against the source text set

## 3.2 Theoretical Basis and Algorithm for Haar Filter

It can be seen that the Haar filter used to calculate DNA plays a very important role for comparison and decision making. Processing through the Haar filter is used multiple times for both the source text and the text to be evaluated. Therefore, in this section we introduce the theoretical basis and develop the algorithm to create DNA sequences for the Haar filter, then the main algorithms will repeatedly use this algorithm as a standard module.

In DWT, the Haar Wavelet line, also known as the Haar filter, is commonly used in time series data mining and indexing [12]. Through research on DWT and Haar lines, to compare the similarity between the two series, we propose the idea of converting text content into real-time series (through a digitizer) and using a Haar filter in DWT to detect unusual patterns, text data is converted and represented into real numbers represented by $x = [x_1\ x_2\ ...\ x_N]$. In this section, we only analyze the theoretical basis and develop the algorithm for the Haar filter to generate DNA used in the similarity detection processing system. In the following section, we will further analyze the preprocessing and processing blocks including digitization, DNA organization and determination of the smallest Euclidean distance.

According to Figure 2, the input signal to the filter is a series of discrete numbers including $N = 2^k$ real numbers. A discrete Haar transformation is performed over K iterations and at the $k^{th}$ iteration (or $k^{th}$ level), with $k = 1, 2, ..., K$; The output signal of the transformation is described as follows:

$$x^{(k)} = \left[x_{low}^{(k)}\ x_{high}^{(k)}\ x_c^{(k-1)}\right] \tag{1}$$

In which, the approximate coefficients $x_{low}^{(k)}$ and detailed coefficients $x_{high}^{(k)}$ are given by the subsampling formula as follows:

$$x_{low}^{(k)} = \left(x_a^{(k-1)} f_L\right) \downarrow_2 \tag{2}$$

$$x_{high}^{(k)} = \left(x_a^{(k-1)} f_H\right) \downarrow_2 \tag{3}$$

with $f_L = [1\ 1]$ and $f_H = [-1\ 1]$ are the low-pass and high-pass filter responses, respectively; $x_a^{(k-1)}$ and $x_c^{(k-1)}$ respectively are the approximation coefficients of the second step (*k-1)* and synthesize the detailed coefficients of the signal chain obtained in the previous steps.

At the initialization point, $x_a^{(0)}$ and $x_c^{(0)}$ is given as follows:

$$x_a^{(0)} = x^{(0)} \tag{4}$$

$$x_c^0 = [] \tag{5}$$

in there, $x^{(0)}$ is the original signal sequence (vector *x* after digitization) and [] is an empty vector. Values:

$$x_a^{(k)} \in^{\sim 1 \times N_a(k)}\ with\ N_a(k) = 2^{K-k}, x_c^{(k)} \in^{\sim 1 \times N_c(k)}\ and\ N_c(k) = \sum_{i=1}^{k} 2^{K-i}$$

*k = 1, 2, ..., K* will be updated according to the following formula:

$$x_a^{(k)} = x_{low}^{(k)} \tag{6}$$

$$x_c^{(k)} = \left[x_{high}^{(k)}\ x_c^{(k-1)}\right] \tag{7}$$

It can be easily proven:

$$N_a(k) + N_c(k) = 2^{K-k} + \sum_{i=1}^{k} 2^{K-i} = 2^K = N \quad (8)$$

*k = 1, 2, ..., K*. Thereby, the signal after K iterations still has the same length N as the original. In essence, different text segments, after being digitized and passed through the Haar filter, will produce number strings carrying characteristic information that can distinguish the level of difference between them. Therefore, the signal sequences behind the filter are called DNAs.

Finally we develop an algorithm according to the above analysis as follows:

Table 1
Algorithm for determining values for DNA sequences

| | ***Algorithm 1: Identify strings DNA*** |
|---|---|
| 1 | **Input:** Set of real numbers. |
| 2 | **Initialization:** The approximation vectors and coefficients are given by the formulas (4) and (5) |
| 3 | **For** k:= 1→ K |
| 4 | Calculate the kth number series according to the formula (1), (2) and (3) |
| 5 | Update the values for the approximation vector and coefficients according to the formula (6) and (7) |
| 6 | **End** |
| 7 | **Output:** The kth number string is the DNA to be calculated. |

# 4 Proposed Algorithms

In this section, we will analyze in detail the tasks of each block during the processing process. Specifically, the preprocessing stage removes special characters and divides the text into segments. Next, the digitization stage in the main processing stage will convert the words in each segment into a typical real number before sending it to the Haar filter to take samples for calculating DNA for comparison. and decided in the next block. To facilitate a detailed analysis of the processes, we will present a common data preprocessing for both the source and evaluation corpus, after which the main processing will be described separately for source and evaluation corpus.

## 4.1 Data Preprocessing

After being collected, the source documents will be sent to the data preprocessor. Here, the system will consider a sentence as the smallest unit that can be copied. Accordingly, segments will be identified based on punctuation marks (.). Then, non-

alphanumeric characters and alphabetic letters such as (! ? , []…) are removed from the segments. In other words, a segment will represent a sentence in the text after removing special characters. These segments will be stored as raw text for the purpose of retrieving results.

The evaluation text is similarly preprocessed and gives a raw evaluation text. This raw evaluation text can be updated by the system as new data for the purpose of expanding the source database for evaluating other documents.

## 4.2 Build a DNA Set for the Source Text

To build the DNA set for the source text, we digitize the words in each segment. However, it can be noticed that each segment collected from the source texts is a sentence. Accordingly, the number of words in each segment will be different, which leads to the length of the real number signal string for each segment being different. Meanwhile, calculating DNA using the Haar filter (*as in section 3.2*) requires that the length of the signal sequences be equal. So we use a window to sample each segment. This sampling window will shift on each segment to extract samples according to a given superposition ratio to the previous sample. To minimize the number of digitization of words in a segment, we will perform digitization across the segment to create a sequence of real signal numbers before sampling the data. The following section is the detailed content of digitizing and standardizing data, sampling for each segment, and organizing data for the source DNA set.

### 4.2.1 Digitize and Standardize Data

In this step, we will digitize each word in the segment into a real number that characterizes that word and distinguishes it from real numbers representing other words. To do this, we first use Unicode to convert the characters in each word into an integer in the decimal system, then concatenate the sequence of these integers in their correct order in the decimal system. a word to form a representative integer. However, the number of digits for each character in the code table is different, so it happens that a string of combined numbers can be made up of many different words. Therefore, we standardize the value for each character by determining a maximum number of digits for each character based on the Unicode code table, this maximum number of digits is called $m$. For example, if a character has a value in the Unicode encoding consisting of $m'$ digits, for $m'<m$, that value will be preceded by $m''$ zeroes so that $m'+m'' = m$. The highlight of this normalization is that the position of the digitized value for each character after string concatenation will be maintained relative to its position in a word. Suppose a certain $i^{th}$ word in the segment has the character $L_i$, in which the $l^{th}$ character ($l = 1, 2,..., L_i$) has the Unicode value of $s_{i,l}$. The integer representing that word in a segment (denoted $s_i$) will be calculated according to the following formula:

$$s_i = \sum_{l=1}^{L_i} s_{i,l} \times 10^{m\times(L_i-l)} \quad (9)$$

However, each word in the segment has a different length, so the integers representing a word will have very different magnitudes. To make the comparison accurate, we normalize the value for a word according to the base 10 logarithmic function. Suppose the maximum length of the number string that a word is capable of forming is $M = m*L_{max}$, with $L_{max}$ is the maximum possible word length. Finally, the real number value representing a word is used as the signal fed to the Haar filter in the next block as determined by the following formula:

$$\mathrm{x}_i^w = M - m \times L_i + log_{10}(S_i) \quad (10)$$

### 4.2.2 Sample Each Segment

Suppose a segment has $w$ words and $i^{th}$ word $(i = 1, 2, \ldots, w)$ characterized by value $x_i^w$ like formula (9). To perform DNA generation for a segment, we perform sampling for that segment. Specifically, a sampling window of length is $N = 2^K$ the signal is used to extract $N$ consecutive real number values in $w$ values of the segment by shifting from left to right. Therefore, a segment can generate many samples, and the samples all have equal length $N$. The window offset can be one or more values. The farther the translation distance is, the computational complexity will be reduced but at the same time the accuracy will also decrease. In short, the input to the Haar filter is a sample in the segment given by the following vector:

$$\mathrm{x} = x^0 = [x_1\ x_2\ \ldots\ x_N] = [x_i^w\ x_{i+1}^w\ \ldots\ x_{i+N-1}^w] \quad (11)$$

These samples will then be fed to the Haar filter to create a set of DNA segments.

### 4.2.3 Organize Data for the Source DNA Set

After performing the steps in sections 4.1 and 4.2, we will obtain a set of DNA for the set of collected documents. We sort the DNA set by the first value of DNA ascending. The purpose of sorting is so that the system can perform binary search. It aims to identify DNA that is similar to that of a sample belonging to a certain segment in the evaluation text. Thereby, we can improve the complexity of the text evaluation algorithm. The reason the first value of DNA can be used as the sorting key is because it is the approximate value or sum of the component values after K iterations. So, at this position if the values of two DNA samples (one belonging to the source text and one to the evaluation text) are the same, the two text samples corresponding to these two DNAs will be the same. However, we need one more step of comparing thresholds before making a decision, which will be discussed in detail later.

Table 2
Source DNA set storage algorithm

| | ***Algorithm 2: Calculate the source DNA set*** |
|---|---|
| 1 | **Input:** Set of collected source documents. |
| 2 | **Initialization:** Length for a DNA (N). |
| 3 | Preprocess, segment, and store source text. |
| 4 | **For** each segment, it is necessary to do: |
| 5 | Segment digitization. |
| 6 | Take samples and determine the DNA group of the segment according to algorithm 1. |
| 7 | **For** each DNA in the group, do: |
| 8 | Binary search on the data warehouse to determine the location of the DNA to be stored so that the first values of the DNA sequences in the entire data warehouse are sorted in ascending order. |
| 9 | Insert DNA into the database at the correct location. |
| 10 | **End For** // End of for loop line 7 |
| 11 | **End For** // End of for loop line 4 |
| 12 | **Results obtained:** The data warehouse has been updated and organized. |

### 4.3 Describe the Similarity Detection Algorithm

After preprocessing the evaluation text, we can easily perform the process of encoding the evaluation text data as in the previous sections. For source documents, their DNA is stored as a database, like a library available to compare similarities. Meanwhile, the evaluation text after being segmented will be put into sequential encoding as a real-time sequence signal. Accordingly, each segment will be sampled into a group of DNA and this group of DNA will be sent to the comparator. Then, group the DNA of the following segment in the same process until the evaluation text is exhausted.

#### 4.3.1 Compare and Make a Decision

In the final block of the system, we will compare each group of DNA segments with the DNA of the stored source data set. For each DNA sample in the DNA group included in the comparison stage, we will do a binary search in the data warehouse to determine which source DNA has the first value that is most similar to the DNA under consideration. Next, the Euclidean distance between two DNAs is calculated very simply according to the following formula:

$$d(x, y) = \|x - y\|_2^2 \tag{12}$$

in there, $x \in^{\sim 1 \times N}$ and $y \in^{\sim 1 \times N}$ are the source DNA vector and the DNA vector under consideration. This Euclidean distance will be compared to a threshold level $\varepsilon$. If $d(x, y) < \varepsilon$, the two DNAs are considered identical and the position

corresponding to the DNA under consideration is marked again for the system to make a decision after synthesizing all DNA samples of the segment. The similarity detection algorithm between two documents is described in Table 3.

Table 3
Algorithm to detect similarities between two texts

| | ***Algorithm 3: Detect similarities between two texts*** |
|---|---|
| 1 | **Input:** Text to evaluate. |
| 2 | **Initialization:** DNA sequence length (N) and threshold level (ε) for similarity comparison. |
| 3 | Preprocess, segment, and store data to export results. |
| 4 | **For** each segment, it is necessary to do: |
| 5 | Segment digitization. |
| 6 | Sample and determine the DNA group of the segment according to Algorithm 1. |
| | **For** each DNA y in the group, it is necessary to do: |
| 7 | Binary search on the source DNA repository to find a DNA x such that the |
| 8 | starting value of the considered DNA sequence y is closest to the starting value of DNA x. |
| | Calculate the Euclidean distance d(x, y) according to formula (11). |
| 9 | **If** d(x, y)< ε then |
| 10 | Mark the DNA y under consideration. |
| 11 | **Endif** |
| 12 | **End for** // End of for loop line 7 |
| 13 | Synthesize the marked DNA (if any) to obtain a sequence of similar words of the segment under consideration compared to the source segment. |
| 14 | **End for** // End of for loop line 4 |
| 15 | **Obtained results:** Provide segments containing words similar to the source segments (if any). |
| 16 | |

# 5 Experimental Results

PAN proposed a copy detection evaluation model based on the source dataset (data warehouse) [13, 14]. To test the results of the proposed algorithm, we use measurements in PAN to calculate the prec (precision) and rec (recall) values. Specifically, we call the set of copied character strings and the set of detected character strings respectively as follows:

$$\mathrm{S} = \{S\} \tag{13}$$

$$\mathrm{D} = \{D\} \tag{14}$$

where, $S$ and $D$ are the source text strings that are copied and the evaluation text strings that are detected to be similar to the strings in the source text, respectively.

The values *prec* and *rec* are determined by the formulas according to [15], which are:

$$prec = \frac{1}{|\mathrm{D}|}\sum_{D\in \mathrm{D}} \frac{|D \cap (\mathrm{U}_{S\in \mathrm{S}} S)|}{|D|} \tag{15}$$

$$rec = \frac{1}{|\mathrm{S}|}\sum_{S\in \mathrm{S}} \frac{|S \cap (\mathrm{U}_{D\in \mathrm{D}} D)|}{|S|} \tag{16}$$

in there, |S| and |D| are the number of elements in the set S and D, $|S|$ and $|D|$ are the length of the string $S \in \mathrm{S}$ and $D \in \mathrm{D}$.

Through 10 times of testing on the PAN data set, each time evaluating 100 suspicious texts that are completely different from the texts used to find the threshold value ε. We set the values as Table 4 and the results are as Table 5 and Figure 3.

Table 4

Set values for the test process

| **Parameter** | **Setting value** |
|---|---|
| Maximum number of digits to encode a character, m | 5 |
| Maximum number of characters in a word, Lmax | 45 |
| Sample length DNA, N | 8 |
| Number of iterations to process with Haar filter, K | 3 |
| Threshold value for Euclidean distance, ε | $10^{-7}$ |

Table 5

Experimental results

| **Test time** | \|S\| | \|D\| | **prec (%)** | **rec (%)** |
|---|---|---|---|---|
| 1 | 6028 | 6004 | 98.12 | 97.73 |
| 2 | 5336 | 5315 | 97.99 | 97.60 |
| 3 | 9340 | 9310 | 98.02 | 97.71 |
| 4 | 6491 | 6455 | 97.86 | 97.32 |
| 5 | 8063 | 8033 | 97.77 | 97.41 |
| 6 | 6982 | 6960 | 97.63 | 97.32 |
| 7 | 6472 | 6460 | 97.77 | 97.59 |
| 8 | 5621 | 5595 | 98.14 | 97.69 |
| 9 | 6519 | 6506 | 97.86 | 97.67 |
| 10 | 6026 | 6005 | 98.00 | 97.66 |
| **Average** | **6687.8** | **6664.3** | **97.92** | **97.57** |

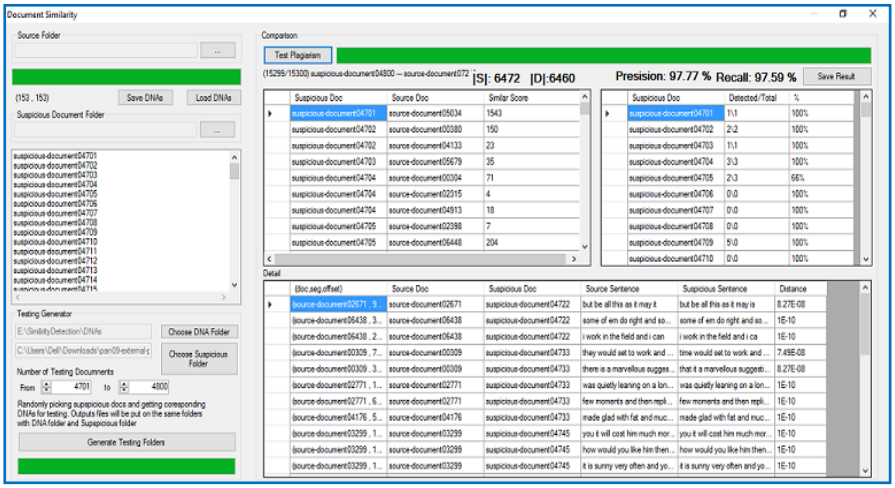


Figure 3
Single-test results interface

With the above results, we see that with different numbers of elements in the two sets (from about 5,300 to 9,300 elements), our proposed algorithm gives very high and stable prec and rec results. determination (more than 97%).

In this experimental part, we choose the PAN 2009 standard data set, which is an artificial data warehouse to evaluate the copy detection system including 41,223 documents with 94,202 plagiarism cases studied by many research groups and laboratories around the world use it to evaluate copy detection methods [15]. Besides, we also use the metrics for evaluation in PAN competitions. Therefore, the results achieved are completely reliable to evaluate the new algorithms and approaches to detect text similarity that we proposed.

## Conclusion

We have proposed a system model, processing process and presented a completely new approach to detecting text similarity, which is based on the DWT method and the Haar filter. The major contribution in the paper is to propose an algorithm to convert text into DNA strings of real numbers, and build an algorithm to detect similarities between texts. Experimental results on the standard PAN data set prove that our proposed algorithm is highly effective in detecting text similarity.

In the coming time, we will continue to research to further optimize the proposed algorithm and test on many other data sets. In particular, improving the algorithm to apply to Vietnamese text brings high efficiency to solve the problem of detecting copying of Vietnamese text.

**Acknowledgement**

This research was funded by the Ministry of Education and Training (Vietnam) through the project code B2022-DNA-17.

**References**

[1] Sebestyén Katalina, Csapó Gábora, Mária Csernochb, Bernadett Aradib, "Error Recognition Model: High-mathability End-user Text Management", Acta Polytechnica Hungarica, Vol. 19, No. 1, pp. 151-170, 2022
http://dx.doi.org/10.12700/APH.19.1.2022.1.10

[2] Peter Coates, Frank Breitinger, "Identifying document similarity using a fast estimation of the Levenshtein Distance based on compression and signatures", Proceedings of the Digital Forensics Research Conference Europe (DFRWS EU), 2022
https://doi.org/10.48550/arXiv.2307.11496

[3] Mohamed El-Rashidy et al., "An effective text plagiarism detection system based on feature selection and SVM techniques", Multimedia Tools and Applications, Volume 83, pp. 2609-2646, 2023
https://doi.org/10.1007/s11042-023-15703-4

[4] Trung Hung Vo, Thi Le Thuyen Phan, Khanh Chi Ninh, "Development of a fake news detection tool for Vietnamese based on deep learning techniques", Eastern-European Journal of Enterprise Technologies – EEJET, Vol. 5, No. 2(119), pp. 14-20, 2022
https://doi.org/10.15587/1729-4061.2022.265317

[5] Trung Hung Vo, "Development of a document classification method by using Geodesic distance to calculate similarity of documents", Eastern-European Journal of Enterprise Technologies – EEJET, No. 106, Vol. 4/2, pp. 25-32, 2020
https://doi.org/10.15587/1729-4061.2020.203866

[6] Saqib Iqbal Hakak et al., "Exact String Matching Algorithms: Survey", IEEE Access, Vol. 7, pp. 69614-69637, 2019
https://doi.org/10.1109/ACCESS.2019.2914071

[7] Ali Amer and Hassan Abdalla, "A set theory based similarity measure for text clustering and classification", Journal Big Data 7, 74, 2020.
https://doi.org/10.1186/s40537-020-00344-3

[8] Jiapeng Wang and Yihong Dong, "A Survey of Text Similarity Approaches", Information, 11(9), 421, 2020
https://doi.org/10.3390/info11090421

[9] Ning Ding et al., "Sentence and Document Representation Learning", Representation Learning for Natural Language Processing, Springer, pp. 81-125, 2023
https://doi.org/10.1007/978-981-99-1600-9_4

[10] Juuso T. Olkkonen (Ed), "Discrete Wavelet Transforms - Theory and Application", InTechOpen, ISBN 978-953-307-185-5, 2011 https://doi.org/10.5772/649

[11] Vidakovic, B., "Statistical modeling by wavelets", A Wiley-Interscience Publication, John Wiley & Sons Inc., 2009

[12] Muhammad Junaid Khalid et al., "Integration of Discrete Wavelet Transform, DBSCAN, and Classifiers for Efficient Content Based Image Retrieval", Electronics, Vol. 9, No. 11, 2020 https://doi.org/10.3390/electronics9111886

[13] Martin Potthast, et al, "Overview of the 5th International Competition on Plagiarism Detection", In CLEF (Online Working Notes/Labs/Workshop - https://pan.webis.de), 2013

[14] Olena Zimbacorresponding and Armen Yuri Gasparyan, "Plagiarism detection and prevention: a primer for researchers", Reumatologia, Vol. 59, No. 3, pp. 132-137, 2021 https://doi.org/10.5114/reum.2021.105974

[15] Kadir Yalcin, Ilyas Cicekli, Gonenc Ercan, "An external plagiarism detection system based on part-of-speech (POS) tag n-grams and word embedding", Expert Systems with Applications, Vol. 197, Elsevier, 2022 https://doi.org/10.1016/j.eswa.2022.116677