JOURNAL OF APPLIED
COMPUTER SCIENCE
Vol. 26 No. 1 (2018), pp. 33-62

The Evaluation of Text String Matching Algorithms as an Aid to Image Search

Joanna Ochelska-Mierzejewska[1]

[1]*Lodz University of Technology*
Institute of Information Technology
ul. Wólczańska 215, 90-924 Łódź, Poland
joanna.ochelska-mierzejewska@p.lodz.pl

Abstract. *The main goal of this paper is to analyse intelligent text string matching methods (like fuzzy sets and relations) and evaluate their usefulness for image search. The present study examines the ability of different algorithms to handle multi-word and multi-sentence queries. Eight different similarity measures (N-gram, Levenshtein distance, Jaro coefficient, Dice coefficient, Overlap coefficient, Euclidean distance, Cosine similarity and Jaccard similarity) are employed to analyse the algorithms in terms of time complexity and accuracy of results. The outcomes are used to develop a hierarchy of methods, illustrating their usefulness to image search. The search response time increases significantly in the case of data sets containing several thousand images. The findings indicate that the analysed algorithms do not fulfil the response-time requirements of professional applications. Due to its limitations, the proposed system should be considered only as an illustration of a novel solution with further development perspectives.*

The use of Polish as the language of experiments affects the accuracy of measures. This limitation seems to be easy to overcome in the case of languages with simpler grammar rules (e.g. English).

Keywords: *text comparison, N-gram, Levenshtein distance, Jaro coefficient, Dice's coefficient, Overlap coefficient, Euclidean distance, Cosine similarity, Jaccard similarity.*

1. Introduction

A search is the act or process of analysing a data set in order to find a specific element. This process is most commonly associated with the search for textual or multimedia information. Recent years have seen great progress in the field of search techniques, leading, among others, to the development of methods based on the analysis of graphical objects or sound. Those include specialised solutions applied in medicine or acoustics.

A key to a successful search is the use of suitable algorithms. Years of research have led to the development of many types of solutions, often depending on the nature and type of search.

1.1. Classification of Text String Matching Algorithms

The text string matching algorithms can be classified according to the following criteria:

1. The method of comparison applied.
2. The number of patterns used.
3. The use of pre-processing or text pattern.

The first criterion distinguishes two methods: *exact string matching* and *inexact* or *approximate string matching*.

Exact string matching is the primary way of comparison applied in all search engines. This type of algorithms return a positive result only if an exact match is found. Any variations of the term that we look for, which are not in an identity relation with it, are ignored. An example of such a solution is the naive algorithm (also referred to as the *brute force algorithm*) or the Boyer-Moore algorithm.

The *approximate string matching* consists in finding the strings that are similar to the pattern, rather than identical to it. The degree of similarity is determined arbitrarily on the basis of the number of transfers needed to transform a string into the pattern, or by the length of the substring of the pattern retrieved from the text. The most popular algorithms in this group include the Hamming and Levenshtein distance algorithm and Seller's algorithm.

Inexact string matching comprises four main categories of algorithms:

1. *Dynamic Programming algorithms*, e.g. the Wunsch algorithm and the Smith-Waterman algorithm.

2. *Deterministic* and *Non-Deterministic Finite-State Finite-State Automaton*.
3. *Bit-Parallelism algorithms*, e.g. the Shift-Or algorithm.
4. Algorithms based on filtering. These include most of the algorithms proposed in recent years.

The second criterion specifies three groups of algorithms:

1. The *single pattern algorithms*. These include the commonly used algorithms, such as the naive algorithm, the Rabin-Karp algorithm, the finite automata algorithms, the Knuth-Morris-Pratt algorithm, the Boyer-Moore algorithm, or the Bitap algorithm, also known as Shift-Or, which, thanks to its efficiency, has become the heart of the Unix search tool – agrep.
2. Algorithms using a finite set of patterns. This group includes the Aho-Corasick algorithm, the Commentz-Walter algorithm and the Rabin-Karp algorithm.
3. Algorithms using an infinite number of patterns. In this case, patterns are represented by grammar or regular expressions.

The last criterion of textual analysis algorithms is concerned with the pre-processing of text and pattern. The algorithms that employ the pre-processing of text were developed on the basis of the so-called *online* algorithms which allow the pre-processing of the pattern exclusively. They employ indexing of the content, which makes the search process much more efficient.

1.2. Performance of Text String Matching Algorithms

The running time of an algorithm is fundamental to all complex computational processes. In principle, text analysis algorithms operate on large data sets. Their expected running time depends on the field of application. However, it should be relatively short. In the case of a web browser, it should not exceed a few seconds, whereas a system analysing terabytes of data in a research laboratory, should return results within ten minutes.

Research into text comparison algorithms has shown dependencies between the running time of an algorithm, the memory consumption and the accuracy of results. Those dependencies can vary, depending on the solution employed. Some algorithms work equally well with patterns of any length, other are effective only

for patterns of a specific length. However, in general, we can state that there is an inversely proportional relationship between the running time of an algorithm and the memory consumption (in the case of algorithms based on finite automata) and between the running time of an algorithm and the accuracy of text strings matching (exact vs. approximate search) [1].

The rest of the paper is organized as follows. Chapter 2 introduces the most significant definitions used in the paper. In Chapter 3, the stages of text analysis are presented and selected word, sentence and text comparison methods are described in detail. Next, in Chapter 4, the evaluation of text comparison methods is presented. Chapter 5 presents the outcomes of experiments conducted with the use of methods described in Chapter 3. Chapter 6 focuses on image search based on multi-sentence queries. This is followed by the discussion of the results. Chapter 7 is a summary of the paper.

2. Fuzzy Sets and Linguistic Variable

The concept of a fuzzy set was introduced by Zadeh in 1965 [2]. To each element of a considered space, he added a positive real number from the $[0, 1]$ interval interpreted as a "membership level (or degree)".

Formally, a fuzzy set A in a non-empty space X is a set of ordered pairs

$$A = \{< x, \mu_A(x) >: x \in X\} \tag{1}$$

where $\mu_A : X \rightarrow [0; 1]$ the *membership function* of A, whose values express the membership level of x in A.

Fuzzy sets are mostly applied to formalize linguistic and imprecise understandable statements which express both properties of objects, e.g. *young* people, *small* house, and amounts, e.g. *very few*, *almost all*.

A linguistic variable [3] is an ordered quintuple $< L, H, X, G, M >$, where:
L is the name of the variable,
H or $H(L)$ is the term-set (linguistic values of L),
X is the universe of discourse,
G is a syntactic rule which generates values (labels) of L,
M is a semantic rule which associates a term from L with a fuzzy set in X.

A linguistic variable is exemplified by: L = "temperature", $H(L) = \{low, medium, acceptable, high\}$, $X = [-40^oC, +40^oC]$, in which M associates to e.g. "high" a non-decreasing monotonic and continuous membership function in X, etc.

3. Text String Matching Algorithms

Text string matching is a complex assessment process that investigates the occurrence of certain features (or lack of those features) in a text string. Here, the term text string refers to any linguistic structure. The hierarchy of algorithms applied in the present paper requires a more precise definition of this term. So, text refers to a language structure which consists of at least one sentence that includes at least one word contained between the marks ".". Because of the heterogeneous character of text, the process of analysis is composed of several stages. In the first stage, the text is partitioned into sentences. Next, each sentence is transformed into a set of words, which constitutes the result of the second stage of the analysis. The elements of collections formed in this way undergo further partitioning, creating for each word a set of tokens that are products of the last stage of this process. Their length is fixed, which has a crucial influence on the final results of the experiment described here. Thus, the tokens ascribe a set of features to a particular word. It follows from the above that the level of similarity between two texts can be described as an overall assessment of similarities between sets of tokens of respective sentences.

Each of the above stages is performed using another group of algorithms. The first of them is a text comparison algorithm that is used for preliminary partitioning of the texts compared and calculation of the sum of partial results. The second group of algorithms transforms the sentences into sets of words and computes individual similarity results. The last group of algorithms, which consists of eight different measures, is used for analysing the level of similarity between particular words on the basis of sets of features.

3.1. Text Comparison Algorithm

The first stage of textual analysis involves the use of text comparison algorithm which directly uses the membership function μ_{RS} (5). The algorithm is based on a fuzzy relation RT defined on a set of texts T. It can be formally defined as follows:

Definition 1 *The fuzzy relation RT on T — the set of all text, is of the form:*

$$RT = \{\langle t_1, t_2\rangle, \mu_{RT}(t_1, t_2) : t_1, t_2 \in T\} \tag{2}$$

with the membership function $\mu_{RT} : T \times T \rightarrow [0, 1]$*:*

$$\mu_{RT}(t_1, t_2) = \frac{1}{N} \cdot \sum_{i=1}^{N(t_1)} max_{j \in \{1,2,\ldots,N(t_2)\}} \mu_{RS}\left(s_i, s_j\right) \tag{3}$$

where:
s_i *— the sentence of number in text* t_1*;*
s_j *— the sentence of number in text* t_2*;*
μ_{RS} *– the value of function defined by (5);*
$N(t_1), N(t_2)$ *– the number of sentences* t_1*,* t_2*, respectively;*
$N = max\{N(t_1), N(t_2)\}$ *— the number of sentences in the longer of the two texts under comparison.*

The first operation performed by this algorithm is the partitioning of the texts into sentences. Then, pairs of sentences are compared by the algorithm and the results are processed according to the formula (3). The similarity coefficient obtained in this way is contained in the range [0, 1], where 0 is interpreted as a total lack of similarity between the texts and 1 — as their identity.

3.2. Sentence Comparison Algorithm

The second stage of text analysis is the comparison of sentences that make up the texts under examination. The pairs of sentences obtained in the first stage are further divided into sets of words. Those sets are compared using word comparison algorithms. The results of the comparison are processed according the formula (7) and constitute the final measure of similarity between the sentences under examination.

Definition 2 *The fuzzy relation RS on S — the set of all sentences, is of the form:*

$$RS = \{\langle s_1, s_2\rangle, \mu_{RS}(s_1, s_2) : s_1, s_2 \in S\} \tag{4}$$

with the membership function $\mu_{RS} : S \times S \rightarrow [0, 1]$*:*

$$\mu_{RS}(s_1, s_2) = \frac{1}{N} \cdot \sum_{i=1}^{N(s_1)} max_{j \in \{1,2,\ldots,N(s_2)\}} \mu_{RW}\left(w_i, w_j\right) \tag{5}$$

where:
w_i *— the word of number in sentence* s_1*;*
w_j *— the word of number in sentence* s_2*;*
μ_{RW} *— the value of function (7);*
$N(s_1), N(s_2)$ *– the number of words* s_1*,* s_2*, respectively;*
$N = max\{N(s_1), N(s_2)\}$ *— the number of words in the longer of the two sentences under comparison.*

3.3. Word Comparison Algorithm

The word similarity algorithms are the last and most important group of text analysis algorithms. Used directly by the sentence comparison algorithm and indirectly by the text comparison algorithm, they have a crucial influence on the efficiency of the whole analysis process. In the present paper, eight different similarity measures have been applied. However, it is possible to replace them with other methods that might be more accurate or less time-consuming. The only condition that they must fulfil is that the results obtained have to be contained in a range [0, 1].

3.3.1. *N*-gram

The first of the word comparison methods applied is an extended version of the *N*-gram measure. The generalized version of this measure analyses substrings of the same and different length. The disadvantage of such an analysis is its high computational cost. This was the reason for introducing a modification that consisted in the elimination of too short, and thus not very representative *n*-grams. These restrictions, referred to as upper and lower limits, reduce the number of comparisons, thus shortening the search response time for words with substantially different numbers of letters. This method, determined by the fuzzy relation *RW*, is defined as [4, 5].

Definition 3 *The fuzzy relation RW on W — the set of all words, is of the form:*

$$RW = \{\langle w_1, w_2\rangle, \mu_{RW}(w_1, w_2) : w_1, w_2 \in W\} \tag{6}$$

with the membership function $\mu_{RW} : W \times W \to [0, 1]$*:*

$$\forall_{w_1,w_2 \in W} \quad \mu_{RW}(w_1, w_2) =$$

$$\frac{2}{(N-n_1+1)(N-n_1+2)-(N-n_2+1)(N-n_2+1)(N-n_2)} \cdot \sum_{i=n_1}^{n_2} \sum_{j=1}^{N(w_1)-i+1} h(i, j) \tag{7}$$

assuming that

$$n_1 \leq n_2 \leq N_{min}$$

where:
n_1 *— the minimal length of sub-sequence of the word;*
n_2 *— the maximal length of sub-sequence of the word;*
$h(i, j) = 1$ *– if a sub-sequence containing i letters of word* w_1 *and beginning from j-th position in* w_1 *appears at least once in word* w_2*; otherwise* $h(i, j) = 0$*;*
$N(w_1), N(w_2)$ *– the number of letters* w_1*,* w_2*, respectively;*
$N = max\{N(w_1), N(w_2)\}$ *– the number of letters in the longer of the two words under comparison;*
$N_{min} = min\{N(w_1), N(w_2)\}$.

The first task performed by this algorithm is to create two sets of tokens for word w_1 and w_2, respectively, each of n_1-length. Next, the terms from the first set are compared with the terms from the second set (in the order of appearance in words). After finding the first match, the search starts again for the next element belonging to the first set. The whole procedure involves sets of tokens of the length contained in the range $[n_1, n_2]$. The result is computed by the formula (7). The final result is calculated by multiplying this value by the fraction representing all possible substrings.

3.3.2. Levenshtein Distance

The Levenshtein distance is one of the most commonly used non-fuzzy methods for measuring text similarity. It determines the level of similarity between words of different length by means of three operations: deletion, substitution and insertion of characters belonging to the compared text strings. The procedure consists in calculating the minimal number of insertions, deletions or substitutions

required to transform the source word into the target word. Each of the above-mentioned operations has an arbitrarily assigned weight, which can also influence the result of the comparison [6].

Definition 4 *Let a and b are sequences of characters with the length m and n;*
a_i is the beginning segment of $a_1, \ldots, a_i$;
$w(x, y)$ is the weight of substitution;
$w(x, \oslash)$ is the weight of deletion;
$w(\oslash, y)$ is the weight of insertion.
Then Levenshtein distance is of the form:

$$d_L\left(a_i, b_j\right) = min \begin{cases} d_L\left(a_{i-1}, b_j\right) + w\left(a_i, \oslash\right) \\ \quad for < deletation\ a_i >, \\ d_L\left(a_{i-1}, b_{j-1}\right) + w\left(a_i, b_j\right) \\ \quad < substitution\ a_i\ with\ b_j >, \\ d_L\left(a_i, b_{j-1}\right) + w\left(\oslash, b_j\right) \\ \quad < insertion\ b_j > \end{cases} \tag{8}$$

The Levenshtein distance is not defined by the membership function. Thus, the results of formula (8) do not belong to the range [0, 1]. Because the function applied in the present paper operates only on values belonging to this range, it is necessary to normalize the obtained result, as illustrated below:

$$dn_L = 1 - \frac{d_L}{max(m, n)} \tag{9}$$

3.3.3. Jaro Coefficient

The Jaro coefficient is used mainly for typographic error and duplicate detection. This algorithm can be also used in statistics for record linkage [6]. According to the experiments conducted by C.D. Budzinsky, this method brought best results in comparison to twenty others used for typographic error finding [7].

Definition 5 *Let us define on the set of all words W a fuzzy relation RWJO of the form:*

$$RWJO = \{\langle w_1, w_2\rangle, \mu_{RWJO}(w_1, w_2) : w_1, ws_2 \in W\} \tag{10}$$

with membership function $\mu_{RWJO}\colon W \times W \to [0,1]$ *given by:*

$$\forall_{w_1,w_2 \in W} \quad \mu_{RWJO}(w_1,w_2) = \frac{1}{3} \cdot \left(\frac{|w_1'|}{|w_1|} + \frac{|w_2'|}{|w_2|} + \frac{|w_1'| - T_{w_1',w_2'}}{|w_1'|} \right) \tag{11}$$

where:
w_1, w_2 *– compared words;*
w_1' *– a string of characters from* w_1*, found in* w_2*;*
w_2' *– a string of characters from* w_2*, found in* w_1*;*

$$T_{w_1',w_2'} = \frac{\textit{the number of transposition of characters } w_1' \textit{ and } w_2'}{2}.$$

In order to extend Definition 5, let us analyze two text strings, w_1 and w_2. Let w_1 contain characters similar to those in w_2 and let w_2 contain characters similar to those in w_1. Character a from string w_1 is similar to string w_2, if it appears in a similar place of string w_2 [8]. According to Jaro, two characters from w_1 and w_2 are considered "matching" if the distance between them is not further than $\frac{max(|s_1|,|s_2|)}{2} - 1$. However, the area of search can be determined arbitrarily.

3.3.4. Dice's Coefficient

The Dice's coefficient is applied in the area of information retrieval. It is a type of measure based on the calculating of terms of compared words. The coefficient is calculated as follows: twice the number of terms found in both strings is divided by the sum of terms found in both words. The formal definition of the Dice's coefficient is presented below.

Definition 6 *Let us define a fuzzy relation RWD between two sets of words X and Y*

$$RWD = \{\langle x,y\rangle, \mu_{RWD}(x,y) : x \in X, y \in Y\} \tag{12}$$

with membership function $\mu_{RWD}\colon X \times Y \to [0,1]$ *represented by:*

$$\forall_{x \in X} \forall_{y \in Y} \quad \mu_{RWD}(x,y) = \frac{2 \cdot \sum_{i=1}^{n} x_i \cdot y_i}{\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i} \tag{13}$$

where:
x, y – compared words;

x_i – *i-term of x;*
y_i – *i-term of y;*
n – *the number of terms in x, y or their common part.*

The above-mentioned terms are usually bigrams. In order to maximize the accuracy of our measurement, it is necessary to divide words into tokens of different lengths. However, this will lengthen the search response time. Here, substrings of lengths contained in the range $[2, \textit{length of the shorter word}]$ were used.

3.3.5. Overlap Coefficient

The overlap coefficient is similar to the Dice's coefficient, both in terms of application areas and ways of comparison. Here, the computations are also based on terms. The overlap coefficient computes the overlap between two words. If they overlap entirely, they are classified as identical.

Definition 7 *Let us define a fuzzy relation RWO between two sets of words X and Y*

$$RWO = \{\langle x, y\rangle, \mu_{RWO}(x, y) : x \in X, y \in Y\} \tag{14}$$

with a membership function $\mu_{RWO} : X \times Y \to [0, 1]$ *represented by the formula:*

$$\forall_{x\in X}\forall_{y\in Y} \quad \mu_{RWO}(x, y) = \frac{\sum_{i=1}^{n} x_i \cdot y_i}{min\left\{\sum_{i=1}^{n} x_i, \sum_{i=1}^{n} y_i\right\}} \tag{15}$$

where:
x, y – *compared words;*
x_i – *i-term of x;* y_i – *i-term of y;*
n – *the number of terms in x, y or their common part.*

Dividing of the compared words into tokens is performed in the same way as in the case of Dice's coefficient.

3.3.6. Euclidean Distance

Another method, which resembles the Dice's measure is the Euclidean distance. It is based on a vector space representing the attributes of compared words. In text analysis, the attributes define the number of identical terms in each of the compared words. The Euclidean distance is formally defined as follows:

Definition 8 *Given two objects X and Y represented by n attributes, where x_i is the i-attribute, the similarity measure $X = [x_1, x_2, \ldots, x_n]$ and $Y = [y_1, y_2, \ldots, y_n]$ expressed by the formula*

$$d_E(X, Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{16}$$

is referred to as Euclidean distance.

The algorithm applied here is based on the comparison of terms. On the basis of the compared words, a and b, we create a set $T_a \cup T_b$ of all (unique) tokens of the length contained in the range $[2, \textit{length of the shorter word}]$. Other elements of the set are then retrieved from the words and the number of occurrences is calculated. Finally, for each of the n-terms contained in the set $T_a \cup T_b$ two attributes are created, namely x_i and y_i. These attributes define the number of occurrences of i-term of set $T_a \cup T_b$ in a and b, respectively. The values obtained in (16), define the distance between the two words.

This distance is not normalized to the range $[0, 1]$. Thus, scaling is performed, using the following formula (17):

$$dn_E = \frac{\sqrt{|T_a|^2 + |T_b|^2} - d_E}{\sqrt{|T_a|^2 + |T_b|^2}} \tag{17}$$

where:
$|T_a|$, $|T_b|$ – the number of tokens of the length from the range $[2, \textit{length of the shorter word}]$ created from words a, b, respectively.

3.3.7. Cosine Similarity

Another commonly used method based on a vector space is the Cosine similarity measure. In this method, the compared words are transformed into vectors. However, as opposed to the Euclidean distance, the similarity level is not determined by measuring the distance between attributes x and y, but by measuring the angle between the vectors. For similar text strings the cosine value of this angle tends to 1, whereas for substantially different strings, whose vectors form an angle of 90^o, it tends to 0. The results of this comparison, contained in the range $[0, 1]$, provide the ability to define a fuzzy relation RWC between the words. The formalized definition of this relation is as follows:

Definition 9 *A fuzzy relation RWC between two sets of vectors X and Y is defined:*

$$RWC = \{\langle x, y\rangle, \mu_{RWC}(x, y) : x \in X, y \in Y\} \tag{18}$$

with membership function $\mu_{RWC} : X \times Y \rightarrow [0, 1]$ *represented by the following formula:*

$$\forall_{x \in X} \quad \forall_{y \in Y} \quad \mu_{RWC}(x, y) = \frac{\sum_{i=1}^{n} x_i \cdot y_i}{\sqrt{\left(\sum_{i=1}^{n} x_i^2\right) \cdot \left(\sum_{i=1}^{n} y_i^2\right)}} \tag{19}$$

where:
x, y – the compared vectors;
x_i – an element of vector x;
y_i – an element of vector y;
n – the number of elements of vector.

The first stage of the Cosine measure algorithm involves constructing unique collections of tokens for words x and y, denoted as T_x and T_y, respectively. The lengths of tokens, like in the other measures, are contained in the range [2, *length of the shorter word*]. Next, a set $T_x \cap T_y$ is created, which is an intersection of T_x and T_y. The size of sets is determined and the values obtained are used in the formula (19), which in this case looks as follows:

$$\mu_{RWC}(x, y) = \frac{\left|T_x \cap T_y\right|}{\sqrt{|T_x| \cdot \left|T_y\right|}} \tag{20}$$

3.3.8. Jaccard Similarity

The last type of method presented in this paper is the Jaccard index. It is most often used to measure the similarity between sample sets. There are many extended versions of this method but the most basic one focuses on the comparison of two objects, X and Y. Each of these objects can either posses or not possess certain features, called attributes. The similarity of X and Y is understood as the comparison between respective attributes. There are four possible combinations that define the number of the attributes:

- $|X \cap Y|$ – attribute occurs in X and Y;
- $|X \cap \overline{Y}|$ – attribute occurs in X but does not occur in Y;

- $\left|\overline{X} \cap Y\right|$ – attribute occurs does not occur in X but occurs in Y;
- $\left|\overline{X} \cap \overline{Y}\right|$ – attribute does not occur in either X or Y.

On the basis of the observed distribution of attributes, Jaccard proposed the following measure of similarity between X and Y.

Definition 10 *Given two objects X and Y, each of them represented by n-attributes belonging to a common domain U, the measure of similarity between X and Y expressed by*

$$RWJ_{XY} = \frac{|X \cap Y|}{|X \cap Y| + |X \cap \overline{Y}| + |\overline{X} \cap Y|} \tag{21}$$

is referred to as the Jaccard index.

The formula (21) can be simplified as follows:

$$RWJ_{XY} = \frac{|X \cap Y|}{|X \cup Y|} \tag{22}$$

The Jaccard method is based on attribute matching and takes into account only the attributes occurring in objects X and Y. Thus, a situation $\left|\overline{X} \cap \overline{Y}\right|$, in which attribute $i \in U$ does not characterize any of the objects, was not taken into consideration in formula (21).

The result of similarity computations of words form formula (21) or (22) is contained in the range $[0, 1]$. Thus, the measure described is perfectly suited for a membership function defining a fuzzy relationship RWJ between X and Y.

4. Algorithm of Textual Analysis Results Classification

Textual analysis algorithms are the focus of the presented system. The results generated by those algorithms are real numbers, which makes them more difficult to interpret than natural language expressions. Thus, an algorithm has been developed to match the obtained similarity results with the corresponding linguistic expressions. The algorithm was constructed using the fuzzy set theory [2]. A linguistic variable search accuracy was defined in order to describe the similarity of a particular query to image description [9]. Linguistic values characterized by its variable constitute a set:

$$T\,(\textit{search accuracy}) = \{\textit{low, medium, high}\}\,.$$

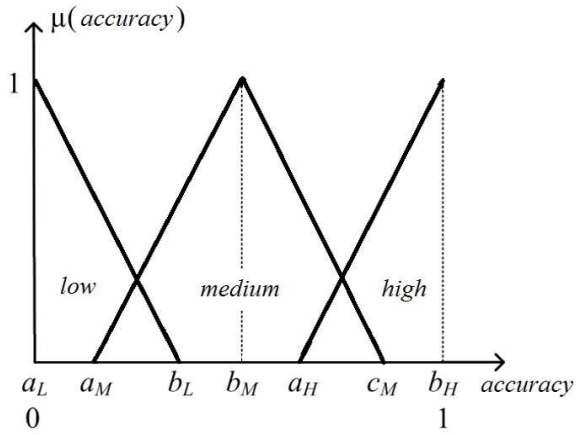

Figure 1: Fuzzy sets for a linguistic variable *search accuracy*.

Each value is assigned to a fuzzy set, the elements of which belong to the range $X = [0, 1]$. A graphic representation of membership functions describing those sets is shown in Figure 1, while their mathematical representation is given by formulae (23), (24) and (25). The type of functions used in the algorithm is chosen arbitrarily. Replacing them with other function types would influence the characteristics of the results but would not influence the method of computing.

$$\mu_L(x) = \begin{cases} 1 & x = a_L \\ \frac{b_L - x}{b_L - a_L} & for \quad x \in [a_L, b_L] \\ 0 & x \in [b_L, b_H] \end{cases} \tag{23}$$

$$\mu_M(x) = \begin{cases} \frac{x - a_M}{b_M - a_M} & x \in [a_M, b_M] \\ 1 & x = b_M \\ \frac{c_M - x}{c_M - b_M} & for \quad x \in [b_M, c_M] \\ 0 & x \in [a_L, a_M] \cup [c_M, b_H] \end{cases} \tag{24}$$

$$\mu_H(x) = \begin{cases} 1 & x = b_H \\ \frac{x - a_H}{b_H - a_H} & for \quad x \in [a_H, b_H] \\ 0 & x \in [a_L, a_H] \end{cases} \tag{25}$$

The boundaries of sets L, M and H are determined by variables a_L, b_L, a_M, b_M, c_M, a_H and b_H, which are directly used from the idea of fuzziness. Such terms as high or low do not possess unambiguous designate. What they refer to is a

subjective feeling, which cannot be unequivocally defined. By substituting specific values with variables, one can arbitrarily define the linguistic values. However, this imposes certain conditions on the sets' boundaries. The fulfilment of those conditions is required by the conceptual model of function μ_L, μ_M, μ_H. There are two inequalities that need to be fulfilled while substituting the variable with certain values:

- inequality for the function μ_L, μ_M, μ_H defined via formula (26)

$$a_L \leq a_M < b_L \leq a_H < c_M \leq b_H \tag{26}$$

- inequality for the function μ_M defined via formula (27)

$$a_M < b_M < c_M \tag{27}$$

The classification of images into sets L, M and H begins with specifying the variables of the membership function. The value $x \in X$ obtained in the process of textual analysis is then compared with the ranges of all sets. If the value belongs to any of the ranges, we calculate the value of the membership function $\mu_L(x)$, $\mu_M(x)$, $\mu_H(x)$, defined on this set. The result of those calculations defines the level of membership of the result of textual analysis in a particular fuzzy set. At the same time, this set carries information about the linguistic value taken by the variable of *search accuracy*.

5. Time Complexity and Accuracy of Text Analysis Algorithms

The most important features that determine the usefulness of a text analysis algorithm are:

- time complexity, understood as the time needed by the algorithm to solve a particular problem;
- accuracy, understood as the quality of a particular solution.

The former signifies the dependency between the number of basic operations performed by the algorithm and the size of the input. It is a necessary but not the only condition for an algorithm to be considered suitable for a particular problem.

However, it can happen that this condition is fulfilled only with a small number of input data and the running time of the algorithm for bigger data sets rises exponentially, which makes the algorithm useless.

The latter feature is less formal. It refers to the level of semantic correctness of text similarity evaluation performed by the algorithm. Due to the specificity of application and quantitative character of computations performed on non-quantitative values, the assessment of the accuracy level is based to a large extent on the judgement of the researcher. However, by evaluating the algorithms in terms of their accuracy, we can indicate the solutions that are unsuitable for textual analysis.

In order to evaluate empirically the usefulness of the algorithms discussed here, it was necessary to conduct certain measurements. In the first step, the methodology consisted in preparing 17 sets $A,\ldots,R$ of random textual sequences of sizes $A = 10$, $B = 20$, $C = 30$, $D = 40$, $E = 50$, $F = 60$, $G = 70$, $H = 80$, $I = 90$, $J = 100$, $K = 110$, $L = 120$, $M = 130$, $N = 140$, $O = 150$, $P = 160$, $R = 170$. Each subsequent set is contained entirely in the previous one, fulfilling the dependency:

$$A \subset B \subset C \subset D \subset E \subset F \subset G \subset H \subset I \subset J \\ \subset K \subset L \subset M \subset N \subset O \subset P \subset R$$

Texts that constitute the elements of those sets were constructed according to an identical pattern (28). The query (29) was built in a similar way. Owing to this unification, it was possible to eliminate discrepancies between the results in the case of phrases of variable length. The change factor results from the random character of constituent units of sentences, i.e. words. They are taken from a specially prepared set of words most often used in the Polish language[1]. The range of their lengths was arbitrarily determined as $[3, 22]$.

$$\text{SENTENCE1.SENTENCE2.SENTENCE3.} \tag{28}$$

where:
the length of SENTENCE1 is 3,
the length of SENTENCE2 is 4,
the length of SENTENCE3 is 2.

$$SENTENCE4.SENTENCE5. \tag{29}$$

[1]The collection of words most often used in Polish contained in file polish.dic.proto (`http://ispell-pl.sourceforge.net`) was developed on the basis of a 3.5 million word set.

where:
the length of SENTENCE4 is 4,
the length of SENTENCE5 is 3.

A text comparison algorithm was used to determine the search response time for each set A,...,R. Thus, the sentence similarity algorithm was used indirectly, together with the N-gram, Levenshtein, Jaro, Dice, Overlap, Euclidean, Cosine and Jaccard measures. The discrepancy between partial results was eliminated by calculating for each set an arithmetic average from 1000 samples by using the same text sequences.

The results are presented graphically in Figure 2, where one can easily distinguish the algorithms that proved most efficient, namely the Overlap, Dice, Jaccard and Cosine measure. In the case of those measures, the average time needed to compare the query with 170 descriptions was not longer than 0.825 seconds. The Jaro algorithm also turned out to be fast, with the running time of 1.325 seconds. In the case of the Levenshtein method, the average measurement time for the same number of comparisons was nearly 2 seconds. The application of the generalized N-gram and Euclidean measure resulted in the highest computing time of 2.892 and 4.068 seconds, respectively.

Average time values shown in linear chart in Figure 2 allow one to clearly determine the efficiency of text similarity algorithm for each measure presented above. Its final value is calculated via formula (30) through the scaling of an arbitrarily selected series of data to the range $[0, 1]$ and calculating for each value its complement to unity.

$$W = 1 - \frac{t_x}{t_{max}} \tag{30}$$

where:
t_x – average running time of text similarity algorithm for a particular measure in series x,
t_{max} – maximal average running time of text similarity algorithm in series x obtained by one of the measures.

The results of these computations are shown in Table 1, whereas the efficiency obtained by each measure is presented in Figure 3.

The present research enables approximate estimation of the running time of each of the above similarity measures in real production environment involving large data sets. Due to linear character of charts, a general formula applied to this

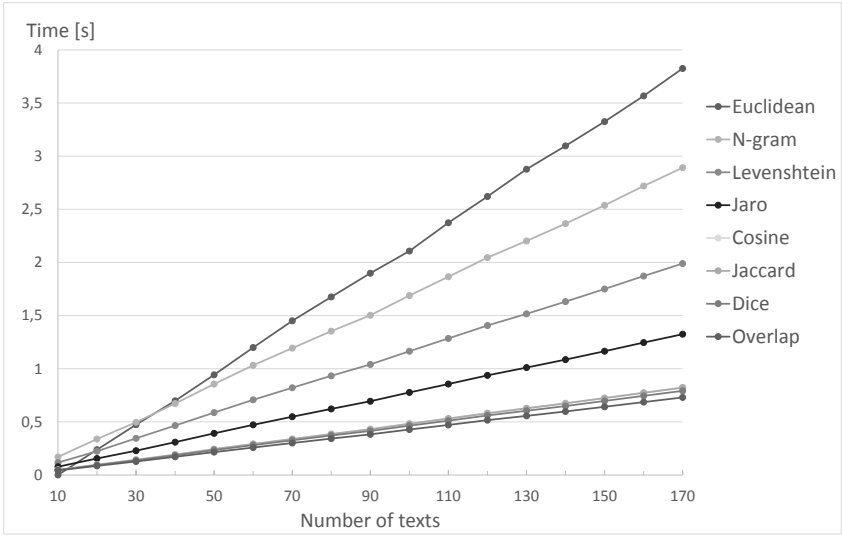

Figure 2: The running time of textual analysis measures – a comparison.

Table 1: Efficiency of text comparison algorithm, depending on the measure applied.

Measure	Mean time of the series 170	Scaled value	Efficiency
Overlap	0.730	0.179	0.821
Dice	0.792	0.195	0.805
Jaccard	0.822	0.202	0.798
Cosine	0.825	0.203	0.797
Jaro	1.325	0.326	0.674
Levenshtein	1.990	0.489	0.511
N-gram	2.892	0.711	0.289
Euclidean	4.068	1.000	0.000

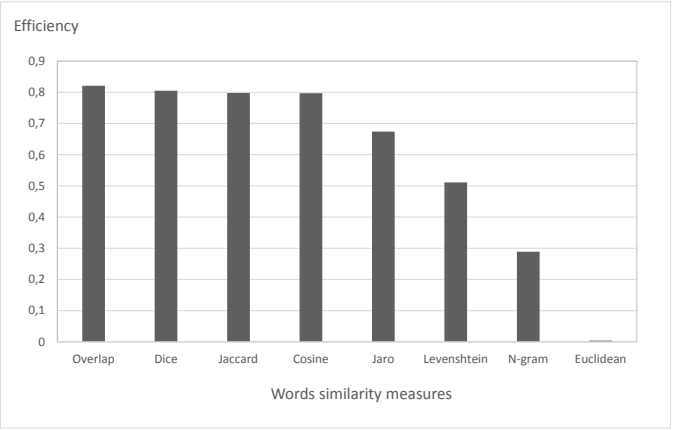

Figure 3: Efficiency of text comparison algorithm for the measures applied.

task will have the form given in (31)[2].

$$y = ax + b \tag{31}$$

where:
a – the slope of a particular function,
x – the number of descriptions for which we calculate the running time of an algorithm,
b – the shift factor of a particular function from Figure 2,
y – the estimated running time of the algorithm for a particular number of texts.

Table 2 presents the estimated running time of algorithms for three sample description sets, calculated via formula (31), with known values of a and b.

In the second stage, we measured the accuracy of the text comparison algorithm. The above-described measures were applied again. The measurement was

[2]The estimation was performed with the method of least squares. It applied a one-equation model with one explanatory variable x. The evaluation of the model was performed using a determination coefficient R^2 and t-Student statistics.

Table 2: An estimated average time of similarity measurement in a production environment

Similarity measure	a	b	$y(10^4)$ [s]	$y(10^5)$ [min]	$y(10^6)$ [h]
Overlap	0.0043	0	42.84	7.14	1.19
Dice	0.0047	0	46.49	7.75	1.29
Jaccard	0.0048	0	48.27	8.04	1.34
Cosine	0.0048	0	48.41	8.07	1.35
Jaro	0.0078	0	90.00	12.97	2.16
Levenshtein	0.0117	0	117.95	19.47	3.24
N-gram	0.0170	0	173.83	28.26	4.71
Euclidean	0.0238	0	239.97	39.73	6.62

based on 15 sets $A,\ldots,O$, each consisting of a query k and two texts h and l.

$$\begin{aligned} A &= \{k_A, h_A, l_A\} \\ &\vdots \\ O &= \{k_O, h_O, l_O\} \end{aligned}$$

For each minimal syntactic change (such as the change from singular into plural, introduction of a participle form, adding a suffix or prefix, or other word formation processes), description h repeated the query, with the semantic identity criterion depending on the subjective opinion of the researcher. In terms of syntax and meaning, text l differed substantially from the query. The lengths of k and h within the set were the same, while the length of description l was arbitrary. A sample set G was as follows (in Polish):

k_G = PRZYKRYTE SNIEGIEM CHOINKI.
ZJEZDZAJACY NARCIARZE.
ZACHODZACE SLONCE.

h_G = POKRYTE SNIEGIEM CHOINKI.
NARCIARZE ZJEZDZAJA.
ZACHOD SLONCA.

l_G = POLNA DROGA.PADA DESZCZ.
CHLOPAK PROWADZI ROWER.
BAGAZNIK PELEN GAZET.

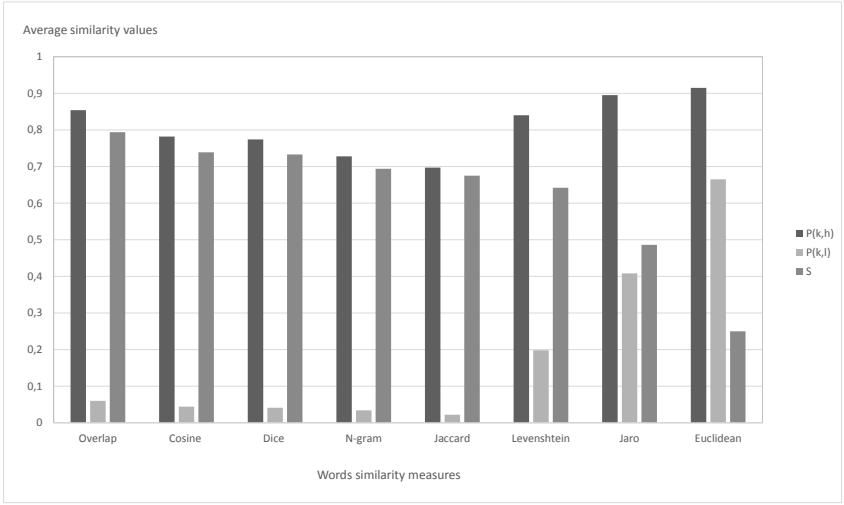

Figure 4: Comparison of average similarity values of t and h, t and l for sets $A, \ldots, O$ and their efficacy S.

Using the above data, two measurements were conducted to determine the similarity between all pairs of texts k and h and k and l. This was performed by calculating for each group a result in the form of an arithmetic average of partial results. The results obtained are presented graphically in Figure 4. For the evaluation of the results, we use a model situation in which the similarity of k and h is a value possibly closest to 1, while the similarity between k and l tends to 0. Thus, the measure of accuracy S is the difference between the two values:

$$S = P_{kh} - P_{kl}$$

As shown in Figure 4, some of the measures used achieved a satisfactory level of accuracy results, with scores at around 0.7. The best result was obtained by the Overlap coefficient, with the value of S equal to 0.794. Almost as good results were achieved by the Cosine method (0.739) and Dice's coefficient (0.733). The N-

Table 3: Usefulness of text comparison algorithm depending on the word similarity measure applied

Measure	Time complexity	Accuracy	Usefulness
Overlap	0.821	0.794	0.807
Dice	0.805	0.733	0.769
Cosine	0.797	0.739	0.768
Jaccard	0.798	0.675	0.736
Jaro	0.674	0.486	0.580
Levenshtein	0.511	0.642	0.576
N-gram	0.289	0.694	0.492
Euclidean	0	0.250	0.125

gram algorithm and Jaccard index, for which the S value equals 0.694 and 0.675, respectively, can also be regarded as efficient. Next come the results obtained by the Levenshtein distance (0.642) and the much less efficient Jaro distance (0.486). The Euclidean algorithm turned out to overstate the level of similarity between k and l, with the average value of S equal to 0.25.

The results obtained by measuring the time complexity and accuracy of text similarity algorithm for each of the presented methods enabled to determine their usefulness for image search systems. A uniform value scale of those two features, simplified the procedure to calculating an arithmetic average. The results were presented in Table 3.

The experiment conducted in the present work allowed the selection of word similarity measures that can successfully be applied in a textual analysis algorithm. The methods were examined in terms of time complexity and accuracy. The final results were presented graphically in Figure 5. Among the eight methods applied, the best result was obtained by the Overlap coefficient (0.807). The Dice, Cosine and Jaccard methods obtained the results of 0.769, 0.768 and 0.736, respectively. The Jaro (0.580), Leveshtein (0.576) and N-gram (0.492) methods were characterized by an average usefulness, with the last of them characterized by a satisfactory accuracy score but very low efficiency. The only algorithm that turned out to be useless was the Euclidean algorithm, with the result as low as 0.125.

Analysing the experiment described, one has to be aware of its approximate nature. Comparing either very similar or extremely different texts or measuring

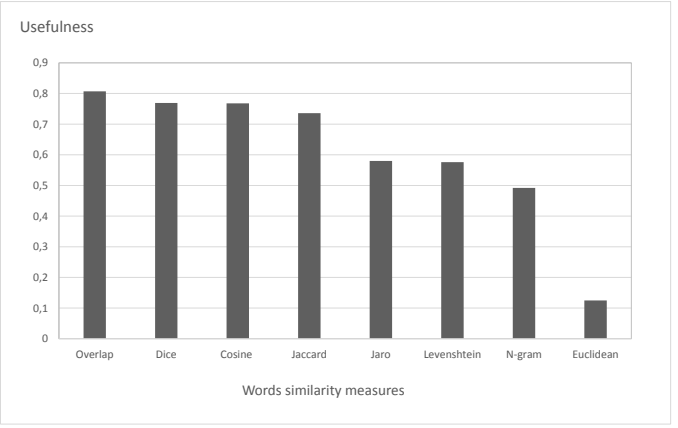

Figure 5: Usefulness of word comparison measures in terms of efficiency and efficacy.

the running time of the algorithm for equal-length sequences is a deliberate operation that aims at showing certain tendencies and does not focus on singular cases. In reality, however, the structure of texts is characterized by different length and different number of constituent elements, which has a direct influence on the computing time[3]. The semantic model of presented in this paper did not take into account the phenomena occurring in natural languages, which could substantially influence the final result. It ignored the existence of synonyms which, despite identical semantic meaning, would not be considered as similar, and homonyms, which are similar only at the syntactic level.

It is evident from the data that there are some dependencies between the time complexity and accuracy of the examined measures. It can be concluded that the running time of an algorithm is inversely proportional to the lengths of compared sequences and the storage space they occupy.

$$time \sim \frac{1}{number\ of\ token}$$

[3]This conclusion was drawn after analysing the influence of words' length on the algorithm's running time.

$$time \sim \frac{1}{storage\ space\ occupied}$$

Every measurement of word similarity is conducted using word-based tokens. Thus, the time required to perform the comparison is determined by the number of tokens. The length of subsequences created is the factor that directly influences the accuracy of word matching, and consequently, the efficacy of the measurement[4]. The number of tokens determines also the memory space required to store them. Thus, the dependency between the time and quality of measurement and the size of storage space becomes obvious[5].

Despite its demonstrative character, the present research can be considered as an important and formally correct frame of reference for more detailed studies on the possible applications of text comparison algorithm and related word similarity measures.

6. Image Search System

Search engines are an integral part of many systems. Without an efficient selection of the data stored, many applications would not be able to provide its users with a basic functionality. The same applies to the system proposed in the present paper, designed to collect and share images. It is aimed to involve an efficient image search engine that will employ algorithms described in Section 3.

The most important element of the system proposed in this paper is the advanced image search engine. It allows the user to define search parameters which are divided into three groups, each controlling other aspects of the search process.

- The first group allows the user to define the ranges of meaning of the terms: low, medium and high, relating to the classification of the images found. The values, which must be included in the interval [0,1], are defined by formulas (26) and (27).
- The second group allows the user to select the algorithms for image search. There are eight methods to choose from, described in detail in Section 3.3.
- The third group allows the user to specify the way in which the algorithm will compare texts. In the case of symmetric variants, the running time will

[4]It was assumed that an optimal length was contained in the range [2, *length of the shorter word*].

[5]The storage aspect was not included in the present discussion, since it does not matter much for the system presented.

be twice longer. The best- and worst-case response times are discussed in Section 3.1.

The results of the search depend on the ranges of meaning defined by the user and the number of algorithms employed. If we define three disjoint sets of meaning, namely *low*, *medium* and *high*, then each of the images will be unambiguously assigned to only one of those sets. However, defining partially joint sets will allow the image to be classified as belonging to each of those sets with a varying membership degree. This will result in the double occurrence of this image on the list of results. As each of the algorithms performs an independent search, the result of the search is the sum of all images found.

6.1. Time Complexity and Accuracy of the Image Search

Verification of the effectiveness of the image search confirmed the results of tests of the text comparison algorithm. A search involving a series of 30 images, described according to their contents, resulted in finding images that matched the specified query. The images with descriptions that were clearly closer to the requested phrase were characterized by a higher degree of membership to a given set, which means a better accuracy of matching as compared to others. Search accuracy depended on the type of algorithm employed and confirmed the results shown in Figure 4. It was also crucial to provide every image with an accurate, intuitively motivated description. Too short a description lowered significantly the chance to find an image, while an exceedingly elaborate description resulted in a lower similarity value and increased the number of false positives.

Table 4 shows the results (expressed as x / y / z, where x – the number images searched by the system, y - the number of correct results, z - the number of medium or high results returned) of four search sessions using all the algorithms described. In each test, a different query was used and only the results from the MEDIUM and HIGH groups were taken into account[6].

The running time of the search engine has remained close to the value presented in the previous section. The differences result directly from the length of the descriptions and length of the query. In most cases, the relative hierarchy remained unchanged, as shown in Table 4. The running times of all algorithms for seven arbitrarily selected images are summarized in Table 5 (the time is given in milliseconds).

[6]Ranges of results: L = [0, 0.25], M = [0.125, 0.875], H = [0, 75; 1].

Table 4: The search effectiveness results

Measure	Series 1	Series 2	Series 3	Series 4
Overlap	4/4/4	3/3/3	3/3/3	3/2/3
Cosine	4/4/4	3/2/2	3/2/2	3/1/2
Dice	4/3/3	3/2/2	3/2/2	3/0/1
N-gram	4/3/3	3/2/2	3/2/2	3/0/0
Jaccard	4/3/3	3/1/1	3/2/2	3/0/0
Levenshtein	4/4/7	3/3/17	3/3/10	3/3/9
Jaro	4/4/30	3/3/30	3/3/24	3/3/27
Euclidean	4/4/30	3/3/30	3/3/30	3/3/30

Table 5: The results of search response time measurement

Measure	Photo 1	Photo 2	Photo 3	Photo 4
Overlap	0.745	1.867	2.023	2.79
Dice	0.859	2.143	2.044	2.931
Jaccard	0.943	2.238	1.529	3.15
Cosine	0.954	4.908	2.214	3.132
Jaro	1.771	3.665	3.029	4.614
Levenshtein	2.444	5.155	3.759	6.877
N-gram	3.058	5.417	4.253	8.568
Euclidean	3.423	7.221	6.709	12.379

Image search in a data set counting several hundred of elements is not a time-consuming process. The time complexity increases noticeably only for data sets of the size characterizing the actual production environment. The results presented in Table 2 reveal the impractical nature of the presented system. The algorithms whose implementation has been optimized for efficiency, have difficulty in dealing with data processing with a speed required by a full-text search. The web application proposed in this paper in its present form is only an example of a system using a novel search engine and does not aspire to be a professional system. It certainly requires some modifications that would reduce the search response time, which in turn would allow the system to be used in a production environment.

7. Summary

Searching for information in the ever-increasing tangle of information is getting increasingly difficult. Systems dedicated to the search for textual information cannot be successfully applied to image search. The new-generation search technologies, such as Clever, Google, or Direct Hit, although ideally suited for multi-word search queries, do not yield equally good results for queries consisting of a few sentences.

The aim of this paper was to develop methods for multimedia objects search, which would provide good efficiency for full-text queries through the use of text analysis algorithms using rudimentary artificial intelligence techniques, such as fuzzy sets and relations. The image search engine proposed in this paper showed a high number of correct hits both for multi-word queries, as well as those consisting of a few sentences. Not all of the applied algorithms coped with this task equally well. A hierarchy of measures was developed on the basis of such parameters as time complexity and accuracy. The shortest search response times were obtained with the Overlap, Dice, Jaccard, Cosine and Jaro similarity measures. The most accurate results were achieved by the Overlap, Cosine, Dice, N-grams and Jaccard methods. The lowest scores in both categories were obtained by the Euclidean distance. A long search response time and overestimated similarity values exclude the application of this measure in effective search mechanisms.

The proposed solution is subject to some limitations. The approximately estimated running times of the algorithms indicate that they do not fulfil the response time requirements of large data sets found in the production environment. In the case of complex resources consisting of thousands of images, the search response

time becomes unacceptably long for professional applications.

The use of Polish as the language of experiments was a natural factor affecting the performance of the algorithms. The varied inflection and thematic alternation, typical of the Polish language, affected word similarity measurements, resulting in a mismatch of semantically identical texts. This limitation will be easier to overcome in the case of languages with simpler grammatical rules (e.g. English) [4].

There are several ways to expand the proposed search engine in order to increase its efficiency. The simplest and most commonly used method is the initial elimination of resources that do not match the query, performed by their thematic categorization. Another solution is to build a mechanism for the translation of descriptions and queries to a unified form, using a common set of words. The translation process would be based on a thesaurus. The third possibility, partly implemented here, is to exclude words which are semantically meaningless, such as conjunctions, prepositions, and pronouns. The elimination of such words would significantly improve the accuracy and reduce the running time of the algorithms.

References

[1] Ng, C. W., *Inexact Pattern Matching Algorithms via Automata http://cmgm.stanford.edu/biochem218/Projects.html*, Tech. rep., Stanford, 2008.

[2] Zadeh, L., *Fuzzy Sets*, Information and Control, Vol. 8, 1965, pp. 338–353.

[3] Zadeh, L., *The concept of a linguistic variable and its application to approximate reasoning (I)*, Information Science, Vol. 8, 1975, pp. 199–249.

[4] Ochelska, J., Szczepaniak, P., and Niewiadomski, A., *Automatic Summarization of Standarized Textual Databases Interpreted in Terms of Intuitionistic Fuzzy Sets*, In: Soft Computing: Tools, Techniques and Application, edited by P. Grzegorzewski, M. Krawczak, and S. Zadrozny, The Academic Press EXIT, 2004, pp. 204–216.

[5] Niewiadomski, A., *Intuicjonistyczne zbiory rozmyte w komputerowym określaniu podobieństwa dokumentów tekstowych*, Ph.D. thesis, Instytut Badań Systemowych PAN, 2001.

[6] Winkler, W. and Thibaudeau, Y., *An Application of the Fellegi-Sunter Model of Record Linkage to the 1990 U.S. Decennial Census, http://www.census.gov/srd/papers/pdf/rr91-9.pdf*, 2004.

[7] Porter, E. and Winkler, W., *Approximate String Comparison and its Effect on an Advanced Record Linkage System, http://www.fcsm.gov/working-papers/porter-winkler.pdf*, 2003.

[8] Cohen, W., Ravikumar, P., and Fienberg, S., *A Comparison of String Metrics for Matching Names and Records, http://www.cs.cmu.edu/ wcohen/postscript/ kdd-2003-match-ws.pdf*, 2002.

[9] Navarro, G., *A Guided Tour to Approximate String Matching*, ACM Computing Surveys, Vol. 33, No. 1, 2001.