Cybernetics and Systems Analysis, Vol. 44, No. 4, 2008

INVESTIGATION OF ACCELERATED SEARCH FOR CLOSE TEXT SEQUENCES WITH THE HELP OF VECTOR REPRESENTATIONS

A. M. Sokolov

UDC 004.032.26

The results of numerical experiments using artificial data are presented. The experiments are designed for testing theoretically derived properties of a randomized scheme for embedding an edit distance into a vector space. Its application to the search for similar texts is also described as applied to the problems of duplicate filtration and spam detection.

Keywords: *edit distance approximation, approximate nearest-neighbor search, neural information technology, spam detection, duplicate detection.*

1. INTRODUCTION

There is a need to compare long character sequences in many applied problems of various object domains. Web search, large-scale document management systems [1], and genetics [2] are examples of such domains.

To compare sequences, one should specify a metric that must be adequate to the corresponding problem and must be efficiently computable. The Levenstein distance (the classical edit distance) [3] often satisfies the first condition. It is defined for symbolic strings x and y as the minimum number of operations of insertion, replacement, and elimination of symbols that are required for the transformation of x into y. These operations are interpreted as operations occurring during mutations and evolution of genes in genetics problems, as some ways of text distortion for unauthorized advertising in the Internet, or as distortions during optical text recognition in document management systems.

The classical algorithm is well known that computes the Levenstein distance for strings of length n and whose complexity equals $O(n^2)$ [4]. In view of large typical values of n and also a great number of strings, the application of this algorithm in the above-mentioned domains is very difficult. Therefore, a computationally efficient estimate of an edit distance [5] is required.

In [6], a method is developed for the estimation of the Levenstein distance on the basis of embedding such an edit distance into a vector space and also an algorithm is proposed for finding approximate nearest strings on the basis of a modification of the scheme of locality-sensitive hashing [7] (LSH) that uses p-stable distributions [8]. However, the proposed methods should be investigated in experiments and applications.

In this article, the results of numerical experiments on checking the theoretical results of the proposed scheme using artificial data are described. The application of the proposed algorithm of approximate determination of nearest strings to the following topical problems is considered: duplicate filtration (in search machines, document management systems, etc.) and detection of spam, i.e., messages that are, as a rule, of advertising character and are massively sent to a great many people (using web-pages, e-mail, etc.) who have not expressed their desire to receive them.

International Scientific-Educational Center of Information Technologies and Systems, NAS of Ukraine and the Ministry of Education and Science of Ukraine, Kiev, Ukraine, *sokolov@ukr.net.* Translated from Kibernetika i Sistemnyi Analiz, No. 4, pp. 32–47, July–August 2008. Original article submitted August 9, 2007.

1060-0396/08/4404-0493

2. EMBEDDING EDIT DISTANCE

Let us describe the essence of deterministic and probabilistic embeddings (proposed and founded in [6]) of the classical edit distance into a vector space and also schemes of searching for approximate duplicates of character strings on the basis of the proposed embedding.

2.1. Deterministic Scheme

For some symbolic string $x \in \Sigma^n$ and also for a parameter $q \in N$, we call an q-gram a substring of length q. We call a vector of the form $v_{n,q}(x) \in (N \cup \{0\})^{|\Sigma|}$ a q-gram vector in which to each q-gram $\sigma \in \Sigma^q$ corresponds a vector element $(v_{n,q}(x))_\sigma \in N \cup \{0\}$ whose value equals the number of occurrences of σ in x. The Manhattan (l_1) distance between these vectors, i.e., the sum of moduli of differences of vector element values is called the q-gram distance [9] and is denoted by $d_q(x, y)$.

We assume that we have a window of width equal to w symbols and two q-gram length values q_1 and q_2, $q_1 < q_2$. A string x is transformed into a vector $v(x)$ by the concatenation of all q-gram vectors

$$v_{i,w,q} = v_{w,q}(x[i, i+w-1]) \tag{1}$$

for $q = q_1, q_1+1, \ldots, q_2$ and $i = 1, \ldots, n-w+1$.

By the distance between such vectors we understand the corresponding q-gram distance and, based on it, we define a new distance between strings $x, y \in \Sigma^w$ as follows:

$$d^{\Sigma}_{w,q_1,q_2}(x, y) = \Sigma_{q=q_1,\ldots,q_2} d_q(x, y). \tag{2}$$

We denote $\Delta q = q_2 - q_1$. Using vectors (1), we define the following distance for strings $x, y \in \Sigma^n$:

$$D(x, y) = \Sigma_{i=1,\ldots,n-w+1} d^{\Sigma}_{w,q_1,q_2}(x[i, i+w-1], y[i, i+w-1]) / ((n-w+1)(\Delta q+1)). \tag{3}$$

Assume that $q_1 = 2w/3$, $\Delta q = \left\lfloor 1/2(-7+(57+16(w-q_1))^{1/2}) \right\rfloor$, $Q = (\Delta q+1)(\Delta q+2)$, and $t = w - \Delta q + 1$. As is shown in [6],

if we have $\mathrm{ed}(x, y) > k_2$, then we obtain $D(x, y) \geq Qt(k_2/(2(\Delta q+1)) - 2)/((n-w+1)(\Delta q+1)) = d_2$; (4)

if we have $\mathrm{ed}(x, y) \leq k_1$, then we obtain $D(x, y) \leq 2k_1[w^2 + (n+1)]/(n-w+1) = d_1$, (5)

where ed(x,y) is the edit distance.

Under the condition $d_2 > d_1$, the smaller the difference between k_1 and k_2, the more exact can be the approximation of $\mathrm{ed}(x, y)$ for a known $D(x, y)$. We put $w = n^\gamma$. Then the growth index w providing the greatest approximation precision is attained when $\gamma = 0.5$. In this case, we have $k_2 = \Omega(k_1 n^{1/2})$, the time of construction of vectors $v(\cdot)$ equals $O(n^{3/2})$, and their dimension equals $O(n^{5/4})$.

The obtained estimates of the time of construction and dimension of obtained vectors in the deterministic scheme are too large to be efficiently used in practice. Therefore, this scheme is randomized in [6] so that it can be applied to the problem of searching for approximately nearest neighbors and is less critical to resources.

2.2. Randomized Scheme

Let a vector $v^i_{w,q_1,q_2}(x)$ be a concatenation of q-gram (from q_1 to q_2) vectors (1) of a substring $x[i, i+w-1]$ of length w, where i is chosen randomly and equiprobably from the set of possible positions of a window of width equal to w: $i = 1, \ldots, n-w+1$. The dimension of the vector $v^i_{w,q_1,q_2}(x)$ equals $\Sigma_{q=q_1,\ldots,q_2} |\Sigma|^q$.

Let φ^i be a random vector of the same dimension as $v^i_{w,q_1,q_2}(x)$ with elements chosen randomly from the Cauchy distribution $p(x) = (\pi(1+x^2))^{-1}$. For the string x, we construct a hash vector of dimension K: $h(x) = (h_1(x), h_2(x), \ldots, h_K(x))$,

where

$$h_i(x) = \left\lfloor ((v^i_{w,q_1,q_2}(x), \varphi_i) + b_i)/r \right\rfloor, \quad (6)$$

r and b_i are real numbers, and b_i is chosen randomly and equiprobably from the range $[0, r]$.

If a sphere $B(q,k) = \{x \in \Sigma^n \mid \mathrm{ed}(q,x) \le k\}$ is specified, then the family $H = \{h: \Sigma^n \to X\}$ (where X is some finite or countable set of values) is called [7] (k_1, k_2, p_1, p_2)-sensitive or simply locality-sensitive if, for any $x, y \in \Sigma^n$ and any independently and equiprobably chosen hash function $h \in H$, the following conditions are satisfied:

$$\text{if } x \in B(y, k_1), \text{ then we have } \mathrm{Prob}[h(x) = h(y)] > p_1, \quad (7)$$

$$\text{if } x \notin B(y, k_2), \text{ then we have } \mathrm{Prob}[h(x) = h(y)] < p_2. \quad (8)$$

In order that the family H make it possible to distinguish between "close" and "distant" strings, the following condition should be satisfied: $k_1 < k_2$, i.e., a close string x must be closer to y than a distant one and, at the same time, we have $p_2 < p_1$, i.e., close strings must cause a collision of hash functions with a higher probability than distant ones.

As is proved in [6], expression (6) is a locality-sensitive function (for $w = n^{2/3}$ and $r = w$, and for q_1, Δq, Q, and t such as those in the deterministic scheme) and, based on it, one can construct a procedure searching for nearest strings using the following general scheme from [9, 10].

Algorithm of searching for the nearest string with the help of LSH. Let there be a base consisting of P strings of identical length n. As a result of the request for $q \in \Sigma^n$, the approximate nearest neighbor should be returned, namely, a string belonging to the sphere $B(q, k_2)$. All strings $x \in P$ are stored in memory cells as follows.

1. By formula (6), for each string x from the base P, L K-dimensional random hash vectors $h^j(x) = (h_1^j(x), h_2^j(x), \ldots, h_K^j(x))$, $j = 1, \ldots, L$, are generated.

2. For each unique hash vector obtained from all the strings of the base $h^j(x)$, $j = 1, \ldots, L$, $x \in P$, a memory cell is created. The total number of cells equals $C \le L \cdot |P|$.

3. Each string of the base P is associated with the cells whose hash vectors are produced by these cells.

To find the approximate neighbor nearest to an arrived string $q \in \Sigma^n$,

(1) L its hash vectors are formed;

(2) the cells are scanned whose hash vectors completely coincide with those formed for q and that correspond to each hash vector $h^j(q) = (h_1^j(q), h_2^j(q), \ldots, h_K^j(q))$, $j = 1, \ldots, L$;

(3) the list S of strings is compiled that are in the scanned cells from the base P. Since the same string of the base P can occur several times in S, this list can be represented in the form of a multiset.

The procedure comes to an end after scanning all the cells corresponding to hash vectors $h^j(q)$, $j = 1, \ldots, L$, or when the size of the list S attains $2L$.

In [6], the values of probabilities p_1 and p_2 are determined and theorem formulated below is proved.

THEOREM 1. For the described construction of vectors, the value of $\rho = \log(p_1 / p_2)$ and, for values of

$$K = \log_{1/p_2} |P|, \quad (9)$$

$$L = |P|^{\rho}, \quad (10)$$

the search algorithm produces (in S with a probability higher than 1/2) a string y such that we have $\mathrm{ed}(x, y) \le k_2$, where $k_2 = O(zn^{1/3} \ln n)$ and z is a parameter that influences the value of k_2 and the values of the probabilities p_1 and p_2. For the experiments described below, the parameters $k_1 = 1$ and $z = 1.01$ were used. The described hashing of a string can also be realized in the form of a distributed neural network representation [10].

Realization of searching for the nearest string with the help of an LSH forest. For a software implementation of the described LSH procedure, a modification of the described LSH scheme is used here, which is called an LSH forest and makes it possible to find nearest neighbors without updating hash vectors with changing the size of the base P or the parameter k_2 [11].

For each $j = 1,\ldots,L$, all hash vectors $h^j(p)$ of all strings p of the base P are stored in the form of a separate prefix tree T_j whose depth amounts to K levels (the depth of its root equals 0 and those of its leaves equal K). The node points of the tree correspond to values of hash vector elements and contain references to the strings whose hash vectors correspond to the paths from the root of the tree to a given node. Leaves of these trees correspond to cells in the original scheme. In particular, if, in hash vectors of two strings, the first k their elements coincide, then the first k nodes along the path from the root of the tree T_j to the leaves corresponding to these strings also coincide.

After arriving a request string q,

— L and its K-dimensional hash vectors $h^j(q)$ are formed;

— for each hash vector $h^j(p)$, the node is found in T_j that corresponds to $h^j(p)$ with the largest number of coincident first elements of these vectors;

— beginning with the nodes found, all L trees are synchronously scanned in the direction of their roots, and the strings p corresponding to the mentioned nodes are added to the resulting multiset S of strings;

— when all the strings whose hash vectors coincide at a given level are added to S, the procedure is repeated for the next (higher) level until the root is reached or $|S|$ exceeds $2L$.

As a result, we obtain the multiset S of strings (candidates for a neighbor that is approximately nearest to q) that are ordered in decreasing order of depths of tree nodes up to which the first elements of hash vectors of the corresponding string and request coincided. By the level of a string from S we will hereafter understand the depth of the last node of the tree at which the first elements of the hash vectors of the request and this string still coincide.

According to Theorem 5.1 from [11], S contains neighbors approximately nearest to the request q with nonzero constant probability (by analogy with Theorem 1).

3. NUMERICAL INVESTIGATION OF THEORETICAL CONSTRAINTS

3.1. Experimental Base RandomStrings

To check the theoretical results, experiments with the deterministic and probabilistic schemes were performed using artificially generated data. Some (random) string (center) q was randomly edited using the operations of insertion, elimination, and replacement. The strings obtained after such an editing form a collection of strings. Since the total number of operations is much smaller than the length of a string and the edit distance cannot be calculated in view of computational complexity, we assume that the edit distance is approximately equal to the sum of numbers of distortions. A similar assumption is made in [12]. In what follows, we will refer to this collection of strings as RandomStrings.

3.2. Deterministic Scheme

Checking theoretical constraints. The experimental investigation of the validity of the fact whether the value of the distance $D(x, y)$ from formula (3) is in interval (4) and (5) was pursued for strings from the RandomStrings base. In Fig. 1, minimum (crosses) and maximum (noughts) experimental values of the quantity $D(x, y)$ are given for each value of the edit distance ed (x, y) for $n = 5000$ and $n = 10000$. The diagram becomes flat near to the greatest possible value of the distance $D(x, y) = [2(w+1) - q_2 - q_1]$ (40 for $n = 5000$ and 58 for $n = 10000$) when, in each window, there are $2(w - q + 1)$ different q-grams for $q = q_1, q_1 + 1, \ldots, q_2$. The theoretical values of upper (4) and lower (5) bounds of $D(x, y)$ are represented, respectively, by continuous and thin dotted lines. As is obvious, the experimental data correspond to the theoretical estimates.

3.3. Randomized Scheme

Checking theoretical constraints. As in the case of the deterministic variant of the scheme, using artificially generated strings of different lengths for the randomized variant, we experimentally check whether the theoretically predicted values of probabilities p_{col} of collision of hash function (6) are within limits (7) and (8). Figure 2 presents the experimentally obtained probabilities of coincidence of the values of hash function (6) for strings of length $n = 1000$ and $n = 3000$ together with the theoretical points of the upper (noughts) and lower (crosses) bounds. As is obvious, these experimental data also correspond to the theoretical estimates.

Choosing the value of *L* and additional filtering. Since necessary resources grow linearly with L, the values of L (9) turn out to be too large [11] in practice. However, the use of smaller values does not guarantee that at least one string y such that we have $\text{ed}(q, y) \le k_2$ will be in the returned multiset S. Subsequent experiments were performed to examine the influence (on the "quality" of the multiset S) of the choice of smaller (than theoretical) values of L and also of an additional filtering of the multiset S (this filtering can presumably eliminate strings for which we have $\text{ed}(q, y) > k_2$) (false positives).

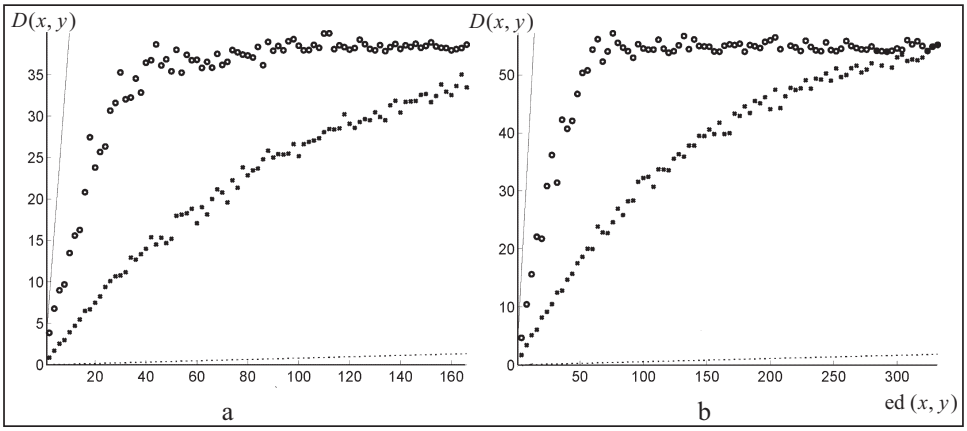

Fig. 1. Typical dependence of the distance $D(x, y)$ on $\mathrm{ed}\,(x, y)$ for $n = 5000$ (a) and $n = 10000$ (b) symbols.

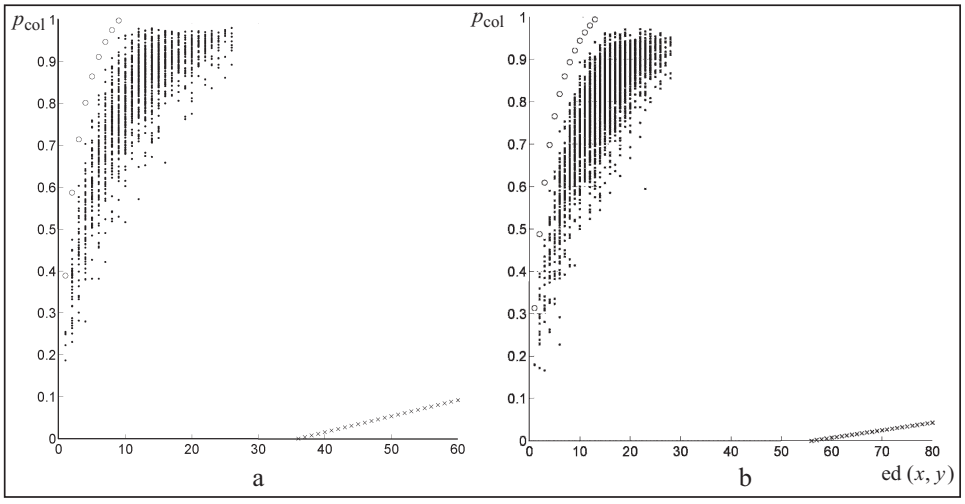

Fig. 2. Dependence of experimental values of the probability of collision of hash functions $h_i(x)$ and $h_i(y)$ on the edit distance between strings of length 1000 (a) and 3000 (b) symbols.

The experiments were conducted using the methods of estimation of quality of the multiset S that are described below.

A. The method of estimation based on the precision value.

B. The method of estimation based on an ordering of the multiset S with respect to a known etalon.

Experiment on the estimation of the quality of S on the basis of the precision value. In search systems for the investigation of quality of the set of documents returned as the answer to a request, a compromise between the number of relevant texts (true positives) and the number of irrelevant texts (false positives) is traditionally analyzed. To this end, the precision value p (the ratio of the number of returned documents relevant to the request to the total number of returned documents) and completeness value r (the ratio of the number of returned documents relevant to the request to the total number of relevant documents) are determined. These values are usually represented by diagrams of dependence of precision on completeness [13].

In the present article, this approach is not informative since the considered problem of searching for the approximate nearest neighbor implies the determination of one neighbor rather that a great many neighbors. Therefore, the ratio (completeness) $|B(q,k_2)\cap P\cap S|/|B(q,k_2)\cap P|$ is not informative.

The situation with the noninformativeness of completeness also arises in estimating the quality of search systems, namely, since the entire set of returned results is of no interest to the user, he usually restricts himself to the browsing of only first pages of the results obtained [14]. In these cases, instead of precision and completeness, the precision characteristic at a level n is usually used (the precision at n / $P@n$) [15] that is computed as the precision on the set consisted of the first n results. In the case being considered, this level is naturally bounded by the size of the multiset S, and the precision at the level $|S|$ is specified as $|B(q,k_2)\cap P\cap S|/|S|$.

TABLE 1

L	Precision Value at the Level $\|S\|$ for									
	$\|S\| = 0.5L$	$\sigma_{\text{р},0.5L}$	$\|S\| = L$	$\sigma_{\text{р},1L}$	$\|S\| = 2L$	$\sigma_{\text{р},2L}$	$\|S\| = 3L$	$\sigma_{\text{р},3L}$	$\|S\| = 4L$	$\sigma_{\text{р},4L}$
1			0.950	0.048	0.945	0.029	0.927	0.025	0.893	0.031
2	0.930	0.065	0.895	0.056	0.853	0.054	0.825	0.032	0.810	0.028
3			0.885	0.050	0.816	0.035	0.784	0.030	0.770	0.025
4	0.855	0.071	0.842	0.041	0.810	0.031	0.777	0.023	0.757	0.021
5			0.824	0.039	0.786	0.021	0.759	0.014	0.735	0.014
6	0.853	0.052	0.797	0.040	0.782	0.025	0.760	0.019	0.730	0.014
7			0.778	0.031	0.768	0.018	0.743	0.013	0.721	0.010
8	0.846	0.038	0.791	0.024	0.755	0.016	0.729	0.013	0.724	0.010
9			0.759	0.024	0.741	0.013	0.717	0.011	0.697	0.009
10	0.860	0.030	0.787	0.024	0.749	0.015	0.722	0.009	0.689	0.008
12	0.807	0.035	0.776	0.021	0.740	0.013	0.712	0.009	0.688	0.007
14	0.821	0.028	0.770	0.018	0.740	0.010	0.716	0.007	0.682	0.006
16	0.809	0.021	0.746	0.013	0.721	0.010	0.691	0.007	0.673	0.005
18	0.845	0.019	0.765	0.014	0.719	0.008	0.686	0.005	0.665	0.004
20	0.811	0.024	0.762	0.014	0.708	0.006	0.682	0.005	0.658	0.004

In order to avoid the introduction of an offset into the precision estimate at the level $|S|$, in view of unequal numbers of strings inside and outside of the sphere $B(q,k_2)$ and different numbers of strings at a definite distance from q, we selected 2200 strings of length equal to 1000 symbols from the RandomStrings collection (Sec. 3.1) (in this case, $k_2 = 126$) in such a manner that the number of strings belonging to $B(q,k_2)$ consisted of half the total number of strings and the number of strings for each value of distance from the center q did not exceed 10. The obtained set P contained strings whose edit distance from the center q varied from 10 to 248.

The set P was stored as an LSH forest with the help of the randomized scheme from item 2.2. A request q was applied to the input of the approximate nearest neighbor search procedure. Based on the set S $(|S| > 4L)$ obtained at the output for each request, the precision at levels $0.5L$ (where L is even), L, $2L, 3L, 4L$ was computed. Precision values were averaged over 100 random independent realizations of the LSH-forest. Their values and dispersion $\sigma_{\text{р}}$ are presented in Table 1.

For small values of L, the maximal precision is observed, which is a distinctive feature of the LSH-forest procedure returning the set of strings S arranged by level depths. With increasing the value of L, the precision first decreases, which is explained by better chances for the strings from $P \setminus B(q,k_2)$ to be in S, but, in what follows, precision ceases to essentially change and the dispersion of its values simultaneously decreases, which allows one to speak about the stabilization of precision values. A similar effect is observed as a result of increasing $|S|$. Thus, in practice, it suffices to use the value of L equal to several tens (for example, 30 or 40). This makes it possible to obtain an acceptable precision level and to save resources. The value of $|S|$ can be fixed, for example, the theoretically recommended value $2L$ can be used.

Experiment on the estimation of ordering of multisets. We investigate the correspondence of string order in S to the actual ordering of the same strings based on the edit distance from q. We denote by S' the multiset of values of the edit distance from the strings from S to q, $S' = \{\text{ed}(x,q) \mid x \in S\}$.

To compare ordered multisets, we will use the generalized Kendall distance [16] as the basis that allows one to compare ordered sets by introducing various penalties for finding elements in different relative orders in sets M_1 and M_2. In particular, if two elements $i, j \in M_1 \cup M_2$ belong to the sets with indices i_1, i_2, j_1, and j_2, then the penalty u is computed according to the following rules:

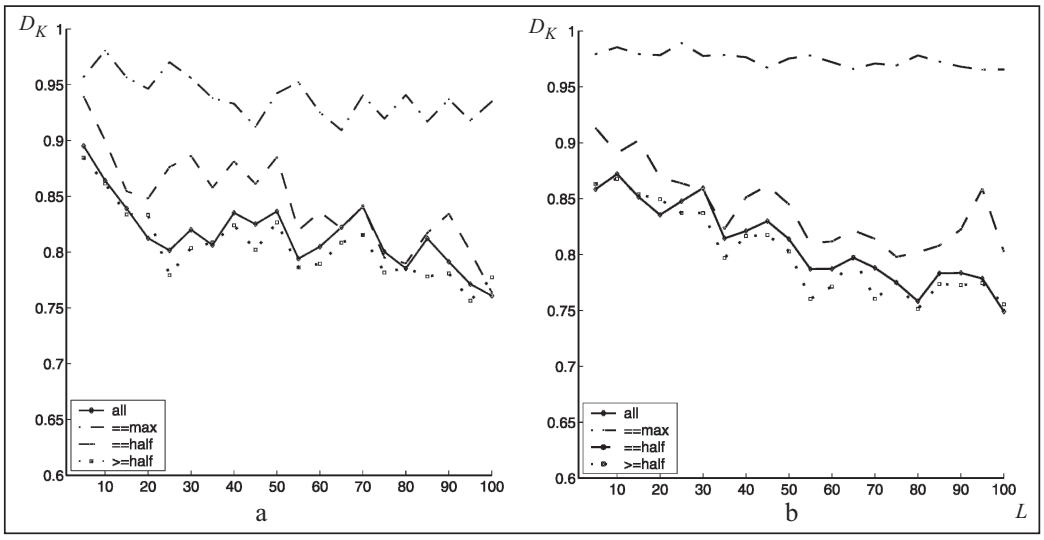

Fig. 3. Dependence of D_K on the number of trees L for the cardinalities of the multiset $|S'|=50$ (a) and multiset $|S'|=200$ (b).

(1) if $i_1 < i_2$, $j_1 < j_2$ $(i_1 > i_2, j_1 > j_2)$, then $u=0$;

(2) if $i_1 < i_2$, $j_1 > j_2$ $(i_1 > i_2, j_1 < j_2)$, then $u=1$;

(3) if $i_2(i_1)$ is not defined, i.e., the corresponding element does not belong to the second (first) set and $j_1 < j_2 (j_1 > j_2)$, then $u=0$;

(4) if $j_2(j_1)$ is not defined, i.e., the corresponding element does not belong to the second (first) set and $i_1 < i_2 (i_1 > i_2)$, then $u=0$;

(5) if $i_1, j_2 (i_2, j_1)$ is not defined, i.e., each element belongs accordingly to its set, then we have $u=p$.

The parameter p is a controllable penalty for a situation in which the first (second) element is absent in one set but is present in the other and vice versa. For these experiments, we used $p=1$.

The distance $D_K(M_1, M_2)$ between sets is the sum of penalties of all pairs of elements $M_1 \cup M_2$,

$$D_K(M_1, M_2) = \Sigma_{i,j \in M_1 \cup M_2} u(i,j). \tag{11}$$

Here, the described algorithm for calculation of the generalized Kendall distance is changed with a view to applying it to ordered multisets (values of edit distances from q to nearest strings) as follows. For each value of the edit distance from $S'_1 \cup S'_2$, entering in at least one of the multisets, its each jth occurrence in both multisets is sequentially transformed into a unique abstract element of a new set and is conditionally denoted by the symbol s_j, where the index corresponds to the serial number under which it enters into this set. In particular, if an element enters in one of the sets only once, then its index is equal to unity. For example, from the sets $S'_1 = \{1,2,3,4,4,3,5\}$ and $S'_2 = \{1,3,3,4,4,3,4\}$, we obtain the following sets of elements: $S'_1 \to M_1 = \{1_1, 2_1, 3_1, 4_1, 4_2, 3_2, 5_1\}$ and $S'_2 \to M_2 = \{1_1, 3_1, 3_2, 4_1, 4_2, 3_3, 4_3\}$. We denote the set of elements that is obtained from S' with the help of the described method by $M(S')$.

Thus, the problem of comparison of the ordered multisets S'_1 and S'_2 is reduced to the problem of comparison of ordered sets $M(S'_1)$ and $M(S'_2)$.

As in the experiment on the computation of the precision at the level $|S|$ (see item A), the number of strings in S that are returned by the LSH-forest procedure and that appears in the description of the algorithm as 2L was variable, i.e., the algorithm returned the mentioned number of strings obtained as a result of ascending LSH trees.

The investigation was pursued on a collection of strings of RandomStrings (see item 3.1). The same 2200 strings of the length equal to 1000 symbols ($k_2 = 126$) as in the previous experiment. The set P was stored as a LSH forest with the help of the procedure considered in item 2.2. A request was applied q to the input of the approximate nearest neighbor search procedure.

We denote by X' the multiset of values of actual edit distances (arranged in increasing order) from the center q to its nearest neighbors, and Y' is the multiset (ordered during the execution of the LSH-forest procedure) of values of edit

distances from the center q to the strings returned by the LSH-forest procedure. We will compute the mentioned distance $D_K(M(X'), M(Y'))$ by formula (11) for $|S'| = 50$ and $|S'| = 200$, where the cardinality of the set $M(X')$ was restricted to the value of $|M(Y')| = |S'|$.

The results were averaged over 100 random independent realizations of an LSH forest. In Fig. 3, the dependence of the distance $D_K(M(X'), M(Y'))$ on the number of trees L in a forest is shown under various constraints on the maximal cardinality of the returned set S' in using the following methods of its additional filtering:

(1) all (without filtering);

(2) == max (only the strings coincident with the request q at the deepest level for a given S);

(3) == half (the strings coincident at the level from $\lfloor K_{\text{avg}} \rfloor$ to $\lceil K_{\text{avg}} \rceil$;

(4) >= half (the strings coincident at a level higher or equal to K_{avg}),

where K_{avg} is the average level value among the returned strings.

As is easily seen, $D_K(M(X'), M(Y'))$ slightly decreases with increasing L, which can be explained by the increase in $|S|$. Therefore, as well as in the previous experiment, one can draw the conclusion about the possibility of fixation of a small value of L.

The filtering methods ==max and ==half take precedence over the other methods and thereby substantiate that the coincidence of strings at deeper levels testifies to their greater similarity.

4. EXPERIMENTS WITH TEXT COLLECTIONS

The method considered in Sec. 3 was applied in the problem of searching for duplicates in text collections and also in the email spam filtering problem. The used approach to the solution of these problems is based on the search for approximate duplicates of text data.

4.1. Problem of Searching for Duplicates in Text Collections

The operation of searching for (approximate) duplicates in text documents requires efficient execution in document management systems. This operation becomes especially required in Internet search machines. The documents that are approximate duplicates with respect to one another are widespread in the Internet. Demonstrative examples are collections of FAQs and guides for programming languages and commands of operational systems. Moreover, some technologies of involving visitors in sites provide for filling "counterfeit" pages by fragments of texts from legitimate pages with a view to displaying paid advertising or redirecting visitors to other sites.

Description of collections. Duplicates were searched for in collections of texts of Reuters-21578 collection [17] (21578 texts of length up to 8316 symbols) and in training texts of the British National Corpus [18] (4054 texts of length up to 2494232 symbols).

The Reuters-21578 collection is a standard collection for investigations in the field of processing text information. The collection contains texts of the Reuters news agency that include many complete and approximate duplicates (abridged versions of detailed news, data on exchange quotation whose dates and numbers in texts are different, etc.).

The British National Corpus (BNC) is a large collection of training texts or samples of modern written and spoken English. It is a "gold standard" and a source of information on "correct" English (in particular, with the help of the collection, the probabilities of prefixes, endings, and words used in different problems are computed). Theoretically, duplicates must be absent in BNC since they distort the statistics of the correct language.

Technique of searching for duplicates. To search for duplicates in text collections, we assume that duplicates are the strings in which there is at least one coincidence of hash vectors of length K . We find duplicates whose number depends on the length n = 100, 150, 250, 500, 1000, 2000 symbols of trimming text and also on the number of duplicates for texts without trimming (equalization according to the maximal length, which is conditionally denoted by $n = 0$). We use the following parameters of the LSH scheme: the number of trees $L = 1, 5$ and dimensions of hash vectors $K = 1, 2, 5, 10, 25, 50, 100, 150, 200$.

Preliminary text filtering was used that preserves only symbols and figures (the *C*-function isalnum ()). The headings of news texts were added to texts, and capital letters were replaced by lowercase ones. Any short text whose length was less than the length of trimming n was extended to the mentioned length by the same special symbol at the end of the text.

Determination of the number of duplicates found in collections. The found number of approximate duplicates in the collections Reuters-21578 and BNC, depending on the values of K and n, is given, respectively, in Tables 2 and 3 for $L = 1$ and $L = 5$. Symbols of the *C*-function isalnum() were used.

TABLE 2

K	Number of Found Approximate Duplicates							
	***L* = 1**							
	n =100	*n* = 150	*n* = 250	*n* = 500	*n* =750	*n* =1000	*n* =2000	*n* =0
1	21347	21334	21281	21224	21206	21213	21275	21458
2	20310	20183	19761	19312	19259	19271	19898	21442
5	8715	7780	5670	5450	7507	10244	15595	20725
10	409	379	806	2124	3273	4721	11109	19199
25	371	350	329	553	1504	1875	4280	17533
50	371	349	325	376	990	1444	3409	15227
100	370	346	322	305	268	271	584	4960
150	370	346	322	305	267	262	479	4823
200	369	346	322	305	267	261	264	2748
K	***L* = 5**							
1	20709	20637	19326	21207	20860	21015	21034	21575
2	19531	19185	19439	19676	19094	20157	20384	21453
5	15513	14532	11858	9593	10393	12436	16536	20658
10	973	1251	2418	4552	6442	8407	14815	20664
25	689	629	615	1832	3168	3769	7238	17401
50	674	600	523	672	1459	2245	4266	14372
100	666	595	513	487	462	505	1592	6497
150	662	592	513	474	437	447	1253	5625
200	661	591	511	467	429	422	591	3499

TABLE 3

K	Number of Found Approximate Duplicates							
	***L* = 1**							
	n =100	*n* =150	*n* =250	*n* =500	*n* =750	*n* =1000	*n* =2000	*n* =0
1	3943	3933	3915	3902	3888	3857	3832	4027
2	3545	3470	3346	3203	3044	2941	2619	4027
5	895	668	297	84	39	31	15	3523
10	10	9	9	8	8	9	9	3447
25	9	9	9	8	8	8	7	3403
50	9	9	9	8	8	8	7	2421
100	9	9	9	8	8	8	7	1263
150	9	9	9	8	8	8	7	1263
200	9	9	9	8	8	8	7	246
K	***L* = 5**							
1	3587	3867	3782	3680	3543	3706	3645	4052
2	3618	3716	3613	3374	3503	3489	3349	4037
5	2187	1855	1020	378	168	81	37	4008
10	12	10	9	8	8	10	18	3997
25	9	9	9	8	8	8	9	3493
50	9	9	9	8	8	8	7	2430
100	9	9	9	8	8	8	7	1738
150	9	9	9	8	8	8	7	–
200	9	9	9	8	8	8	7	–

The number of approximate duplicates naturally decreases with increasing K and is stabilized for large K on values that approximately correspond to the number of duplicates found by the method considered in [19] (320 duplicates). A great number of approximate duplicates in the case when fixed-length trimmings not used (and their increase for the experiment with symbols of isalnum () for $n = 2000$) is explained by a high similarity of many strings since identical symbols has been

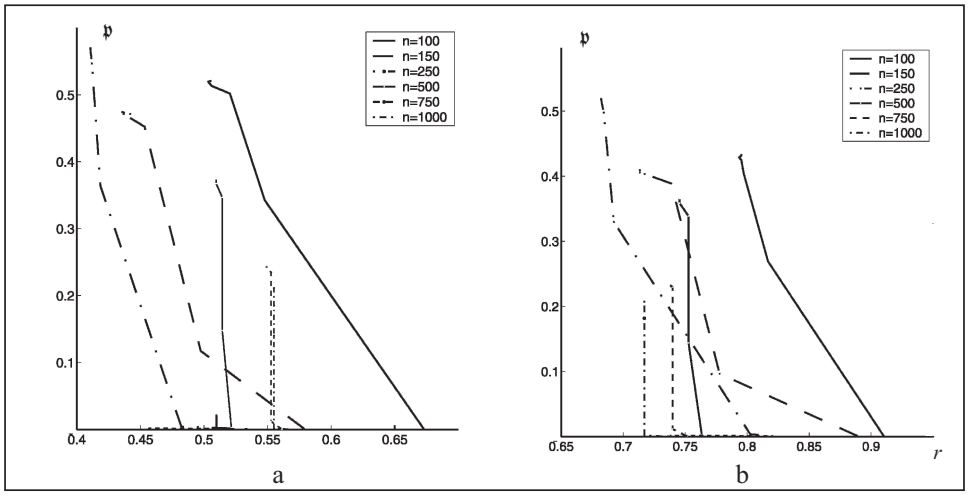

Fig. 4. Dependence of the precision p on the completeness r for the values of $\mathrm{sim} = 0.95$ (a) and $\mathrm{sim} = 0.99$ (b) when $L = 5$.

added to them to equalize their lengths. With increasing L, the number of duplicates also increases since the probability of coincidence of K elements increases in at least one tree. However, as a result of a visual check, some duplicates have turned out to be exact. A decrease in the number of duplicates with increasing the length of trimming for $K \geq 10$ during a visual check is caused by insignificant misprints in texts. For smaller values of K, these misprints are not necessarily cause mismatches of hash vectors.

The time of searching for all duplicates equals the time of traversal of all the leaves in the corresponding LSH tree and does not exceed 0.2 sec on a usual computer AMD Athlon XP 2600 with 1.5 GB of memory.

Comparative results of searching for duplicates. The results of searching for duplicates were compared with those of the deterministic embedding method described in [21]. As the "gold standard," pairs of duplicates were chosen for which the value of the function PERL String::Similarity (based on the edit distance) is no smaller than 0.85 (the same approach was used in creating a collection of duplicates in [20]).

We denote by T_{sim} a set of texts for which the value of the function String::Similarity is larger or equal to sim. Successively assuming the sets $T_{0.95}$, $T_{0.99}$ in the capacity of the "gold standard," one can estimate the quality of our method with the help of diagrams precision–completeness by a modification of the value of K (the values of $K = 5$, 10, 25, 50, 100, 150, 200) were used. As is obvious from Fig. 4, with increasing the threshold T_{sim} by the value of the function String::Similarity, the completeness increases, i.e., a larger portion of "correct" duplicates is added to the multiset S, reaching unity when $\mathrm{sim} = 1$ (only complete duplicates are considered). The decrease in precision with increasing the length of trimming n is explained by a more frequent "successful application" of the method to a large number of special symbols with the help of which text lengths were equalized.

Similar diagrams were constructed in Fig. 5 for the method of deterministic embedding (BY) described in [21] for the lengths of trimming $n = 100$, 150, 250, 500, 750 (for larger values of n, we failed to obtain results in acceptable time). To construct diagrams, we changed the threshold for Hamming distances between the obtained vectors and its value that determined whether it can be considered to be a duplicate. Only the texts were checked in which the Hamming distance did not exceed 15% of that maximally possible for a given length n.

To compare pairs of values precision–completeness, the integral estimate F-measure was used for which $\alpha = 1$ and that was specified as $F_\alpha = (1+\alpha) rp / (\alpha p + r)$ [22], where r denotes completeness and p denotes precision. For each curve depicted in Figs. 4 and 5, the maximal value of $F_{1\max}$ obtained at points of this curve was computed. The obtained values of $n = 250$ and $n = 500$ are presented in Fig. 6. The points (crosses) presented in the legend of BY correspond to the algorithm from [21] and the pluses correspond to the search algorithm using an LSH forest.

The results demonstrate the absence of essential distinctions between the two methods being compared with respect to the "gold standard," but the times of solution of the problem are essentially different. The order of search time for duplicates over the entire collection with allowance made for the construction of trees in the method based on the application of an LSH forest amounts to minutes, whereas this time varies from hours to days in the case of using the deterministic method BY from [21].

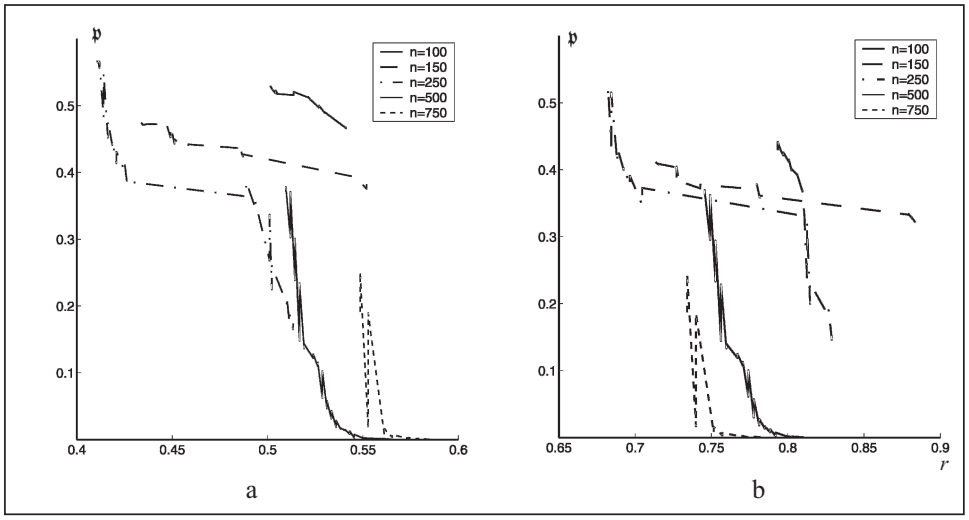

Fig. 5. Dependence of the precision p on the completeness r for the method BY when $\mathrm{sim} = 0.95$ (a) and $\mathrm{sim} = 0.99$ (b).

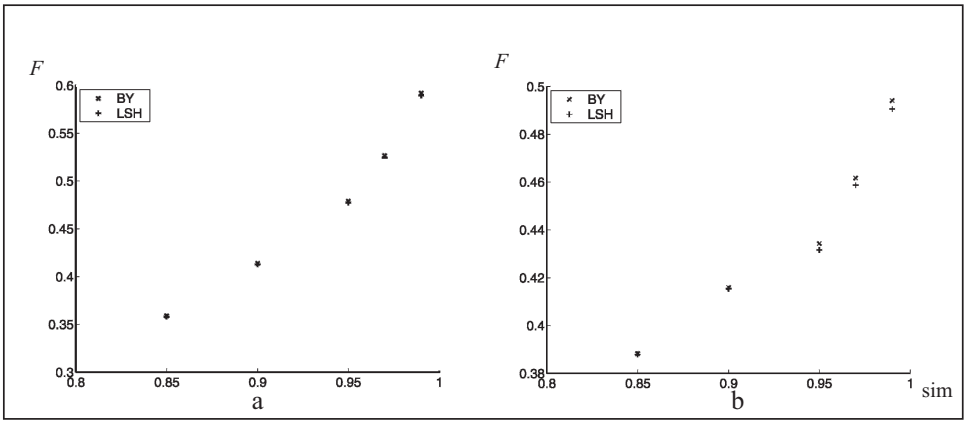

Fig. 6. Values of $F_{1\max}$ for $n = 250$ (a) and $n = 500$ (b).

We also investigated the quality of searching for duplicates using the standard base "Duplicates of Web-pages of the collection ROMIP" [20] (granted by the company "Yandex") that contains a list consisting of more than 10 million pairs of Web-pages whose similarity is no smaller than 0.85 according to the value of the function PERL String::Similarity. The above modification of the method of searching for duplicates (the search for duplicates was performed only among the documents whose length is approximately equal to the length of a request). The values of the estimate F-measure up to 0.88 are obtained depending on the threshold for the value of the function PERL String::Similarity and the parameters K and L.

4.2. Problem of Estimation of the Amount of Spam in Collections of Electronic Letters

The detection of email spam is the urgent problem since the amount of spam among all email messages of an average user has already attained 80–85% in 2005 [23] and continues to increase. Spam is detected by various methods and its detection basically consists of checking spam letters for specific distinctive features of spam letters such as the presence of the address of a sender in a blacklist, the use of HTML marking in a letter, and suspicious enclosures. Bayesian classification is also used [24].

A widespread spam technology that makes it possible to overcome simplest frequency filters is a modification of the text of a letter (for example, a specific way of writing of words that cease to be exact copies of one another and distort the picture of probabilities of words or prevent the preliminary processing of letters by filters [25]). To struggle against such technologies, some researchers propose to detect spam using the comparison of letters with earlier saved ones [26]. We investigated a realization of this idea on the basis of the developed methods of direct comparison of text letter strings. Thus, we estimated the amount of spam that can be detected only with the help of comparison with earlier received spam letters without use of specific knowledge and a detailed analysis of spam technologies.

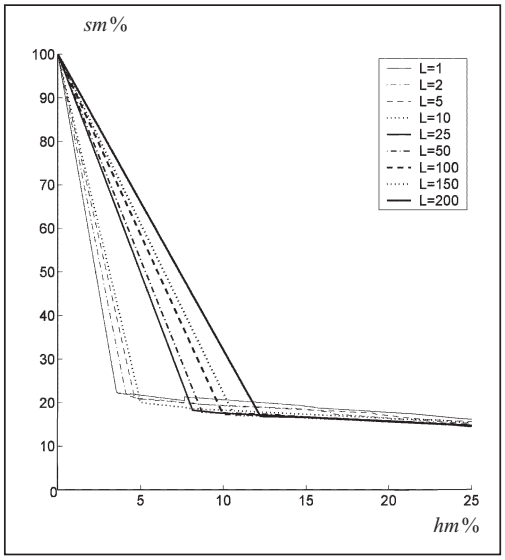

Fig. 7. Dependence of the value of *sm*% on *hm*% for the collection TRACK 2006 Spam Track with assigning score according to the maximum level. The length of $n = 1000$.

Description of used collections. The test base TREC 2006 Spam Track [27] was used, which is widely applied by developers of spam detection systems in testing their products. It contains email messages marked by experts as spam and as nonspam. The English-based part of the base TREC 2006 Spam Track contains $|P| = 37822$ marked real email messages whose size amounts to 189 Mb and that include 24912 (66 %) spam messages.

Scheme of experiments and efficiency estimation. Collections in TREC and the used method of estimation of filter efficiency are constructed with allowance made for the way of real use of spam filters by end users. The destination of filters being tested is to classify messages applied in chronological order and then to be additionally trained as a result of ranging each message in its correct class by the expert. This process corresponds to the usual order of actions of a user when he successively chooses spam from incoming messages and thereby makes it possible to more exactly tune a filter.

According to the conditions of TREC, to each letter must be assigned a parameter, namely, the degree of "spamness" of the letter (score) according to which it is classified as follows: the messages whose score is larger than a definite threshold are considered as spam and the others are considered as usual letters. The quality of operation of a filter is estimated from two key parameters, namely, the percentage of incorrectly classified spam messages *sm*% (false positives) and the percentage of incorrectly classified nonspam messages *hm*% (false negatives). Changing the threshold for the value of score, one can construct ROC curves (the dependence of *sm*% on *hm*%) based on which various algorithms are compared in TREC.

Experiments and assignment of score. A prefilter remained only alphabetic characters (*C*-function isalpha ()) in letters and trimmed messages according to the size $n = 1000$. Messages from collections were applied in chronological order as requests to the approximate nearest-neighbor search procedure with the help of an LSH forest. The value of *L* varied from 1 to 200. The value of *K* was fixed according to formula (9) of the LSH scheme for given p_2 and *P* (see Sec. 2).

Based on the experiments considered in Sec. 3, the following two ways of assigning score to letters are chosen according to their level in the LSH forest to which belongs approximate nearest neighbors of the input letters: according to the maximum level $k_{\max} = K$ and according to the average level k_{avg}.

If a message was actually marked by the expert as "spam" in a collection, then it was added to the set *P* and the next message was applied for classification.

Results. Figure 7 presents ROC curves for the method of assigning score depending on the maximum level (the method based on the mean level yields similar results). As is easily seen, at the level $hm\% = 5 \div 10\%$, approximately 80% of spam messages for the Spam Track 2006 are successfully detected.

As is experimentally established, after realizing the idea of spam detection based on approximate duplicates, its significant part can be filtered, which allows one to estimate the applicability of this approach in large email servers as a

component technology (a module) in more complex systems. The higher the centralization of an email service and the larger the number of its users, the larger percentage of filtered spam can be obtained.

As is seen from ROC curves in Fig. 7, when $L = 1$, the obtained values of *hm*% are often smaller than when $L > 1$. This is explained by a high degree of similarity of a part of legitimate letters to spam, for example, in view of using the html format that remains in a letter in contrast to real spam detection systems in which html codes are usually analyzed and/or eliminated.

5. CONCLUSIONS

The experiments performed have confirmed the efficiency of the method of embedding an edit distance into a vector space and also of a randomized method based on it as methods of finding approximately nearest strings [6]. The possibility of applying the randomized method in real practical problems of searching for duplicates and detecting spam is demonstrated. Taking into account that, in solving these problems, information on the specificity of an object domain is intentionally ignored (although such a specificity must necessarily be used in solving real-world problems in practice), the proposed method showed good results in solving the problem of estimating the amount of email spam and can be used as a component part in spam detection systems.

We assume that systems based on the detection of spam as approximate duplicates will be especially efficient in large email services of the type of Gmail. Since spam is always distributed among a large number of addressees, it might be supposed that it can be detected much easily in an email service than in email boxes of individual users in which the amount of spam is much smaller. A similar approach can be applied to individual users by passing to cooperative spam detection. An example is the distributed spam detection system Vipul that uses sketches of messages marked as spam by the participants in the system. Our hash vectors can be considered as such sketches.

Later on, we supposedly apply the approach to the search for approximate duplicates on the basis of the proposed method of embedding an edit distance to other practical problems such as the search for genes or analysis of logs in computer systems.

REFERENCES

1. S. Brin, J. Davis, and H. Garcia-Molina, “Copy detection mechanisms for digital documents,” Proc. SIGMOD, 398–409 (1995).
2. D. Gusfield, Algorithms on Strings Trees and Sequences, Cambridge University Press, Cambridge (1997).
3. V. I. Levenstein, “Binary codes capable of correcting deletions, insertions, and reversals,” Dokl. Akad. Nauk SSSR, **163**, No. 4, 845–848 (1965).
4. T. K. Vintsyuk, “Recognition of words of oral speech using dynamic programming methods,” Cybernetics, No. 1, 81–88 (1968).
5. P. Indyk, “Open problems,” in: Jiri Matousek (ed.), Workshop on Discrete Metric Spaces and Their Algorithmic Applications, Haifa, Israel (2002).
6. A. M. Sokolov, “Vector representations for efficient comparison and search for similar strings,” Cybernetics and Systems Analysis, No. 4, 18–38 (2007).
7. P. Indyk and R. Motwani, “Approximate nearest neighbors: Towards removing the curse of dimensionality,” Proc. of 30th STOC, 604–613 (1998).
8. M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni “Locality-sensitive hashing scheme based on *p*-stable distributions,” in: 20-th Symposium on Computational Geometry (2004), pp. 253–262.
9. E. Ukkonen, “Approximate string-matching with *q*-grams and maximal matches,” Theor. Comput. Sci., **92**, No. 3, 191–211 (1992).
10. A. Sokolov, “Nearest string by neural-like encoding,” in: Proc. XI-th Conf. Knowledge-Dialogue-Solution, Varna, Bulgaria (2006), pp. 101–106.
11. M. Bawa, T. Condie, and P. Ganesan, “LSH forest: Self-tuning indices for similarity search,” in: Proc. 14th Conf. on WWW, ACM Press, New York (2005), pp. 651–660.

12. S. Azenkot, T.-Y. Chen, and G. Cormode, "An evaluation of the edit-distance-with-moves metric for comparing genetic sequences," DIMACS Technical Report 2005-39 (2005).
13. R. Baeza-Yates and R. Neto, Modern Information Retrieval, ACM Press Series-Addison Wesley, New York (1999).
14. A. Spink, J. Bateman, B. J. Jansen, "Searching the Web: Survey of EXCITE users," Internet Research: Electronic Networking Applications and Policy, **9**, No. 4, 117–128 (1999).
15. D. Hawking, E. Voorhees, N. Craswell, and P. Bailey, "Overview of the TREC8 Web Track," in: 8th Text REtrieval Conference, Gaithersburg (1999).
16. R. Fagin, R. Kumar, and D. Sivakumar, "Comparing top *k* lists," SIAM J. on Discrete Mathematics, 134–160 (2003).
17. Reuters-21578, www.daviddlewis.com/resources/testcollections/reuters21578.
18. The British National Corpus, www.natcorp.ox.ac.uk.
19. M. Sanderson, "Duplicate detection in the Reuters collection," Technical Report (TR-1997-5), Department of Computing Science at the University of Glasgow, Glasgow, UK (1997).
20. Data sets of the competition "Internet-Mathematics," Yandex, http://company.yandex.ru/ grant/datasets_description.xml. 2007.
21. Z. Bar-Yossef, T. S. Jayram, R. Krauthgamer, and R. Kumar, Approximating Edit Distance Efficiently," in: Proc. 45th IEEE Symposium on Foundations of Computer Science, IEEE (2004), p. 550–559.
22. C. J. Van Rijsbergen, Information Retireval, Butterworths, London (1979).
23. "Email Metrics Program: The Network Operators' Perspective," Messaging Anti-Abuse Working Group, Report No. 1, 4th Quarter (2005).
24. P. Graham, Plan for Spam, www.paulgraham.com/stopspam.html (2002).
25. J. Graham-Cumming, "The Spammers' Compendium," in: Spam Conference at MIT (2003), www.jgc.org/tsc.html.
26. A. Kolcz, A. Chowdhury, and J. Alspector, "The impact of feature selection on signature-driven spam detection," in: Proc. 1st Conf. on Email and Anti-Spam (2004), www.ceas.cc/papers-2004/147.pdf.
27. G. V. Cormack, "TREC 2006 Spam Track Overview," in: Proc. 15th Text REtrieval Conf., NIST, Gaithersburg, MD (2006).