APPROACHES TO SEQUENCE SIMILARITY REPRESENTATION

Artem Sokolov and Dmitri Rachkovskij

***Abstract**: We discuss several approaches to similarity preserving coding of symbol sequences and possible connections of their distributed versions to metric embeddings. Interpreting sequence representation methods with embeddings can help develop an approach to their analysis and may lead to discovering useful properties.*

***Keywords**: sequence similarity, metric embeddings, distributed representations, neural networks*

***ACM Classification Keywords**: I.2.6 Connectionism and neural nets, E.m MISCELLANEOUS, G.2.3 Applications*

Introduction and Background

In various applications, it is necessary to search for similar sequences of data. Examples include (but are not limited to) gene similarity search in biology, speech recognition, document database or Internet search, comparison of network traffic flows in computer security systems or detecting dangerous deviations from normal behavior of users by observing sequences of their actions.

There are many ways to formalize an intuitive notion of similarity ("looking the same") between strings[1], e.g., by edit distance. Given a set of edit operations, edit distance $L(s,t)$ between string s and t is the minimum number of edit operations needed to transform s into t. This definition benefits from being intuitive clear, simple to understand, and is often motivated by applications (e.g. evolutionary arguments in biology).

The simplest edit distance is the Hamming distance that counts the number of positions where strings differ – or, alternatively, the number of character changes needed to transform one string to another. Being simple (however, very useful in some applications, e.g. error correction), it is not sufficient for many real-world symbol sequences.

A more general definition is the Levenshtein distance [Levenshtein, 1965], where operations are symbol changes, insertions and deletions. Its exact value can be computed using dynamic programming algorithm in $O(n^2)$ time. More flexible definitions extend the set of possible operations with block copies, moves, indels etc.

In applications, sequence length can be very large, especially in Internet and networking applications and/or applications working with streaming data. Estimating their similarity requires fast algorithms, making time consumption of exact algorithms prohibitive. For example, finding Levenshtein distance in quadratic time is not fast enough. Finding minimum sequence of block edit operations is substantially harder – some versions of it were proved to be NP-hard [Lopresti, Tomkins, 1997].

As representations allowing for a simple (element-wise) definition of similarity or distance, consider vector representations of symbolic sequences. Let each element of a vector represent some item - e.g. some substring of the input string. Depending on the chosen sets of substrings, we obtain representations known by different names in literature. If items are all substrings of nearby symbols of length q and the vector contains their occurrence numbers/frequencies, they are known as "q-grams" [Ukkonen, 1992]. If the sequence is a text and items are single words (i.e., q=1), we get a common “bag-of-words” representation often used in Vector Space Models (VSM) for informational retrieval [Salton, 1989]. The latter case (q=1) does not take order of items into account, but this can be regulated by setting q>1.

Such vector representations can be linked with edit distance through an observation that when two strings s and t are within a small edit distance of each other, they share a large number of items. Vector representations of strings can be used for (weakly) approximating edit distances.

A way to solve the problem of fast finding similar sequences is to look for approximate solutions. One research area where such approximate solutions are sought is embedding techniques [Indyk, 2004]. They deal with

[1] We consider sequences composed of symbols from finite alphabet Σ (e.g., letters, commands), i.e. strings, as opposed to other types of sequences (e.g. speech, movements), where components are not so evident.

mapping complex data to some “easier” space, which allows finding or approximating distances faster. “Easy” spaces are usually vectors spaces; particularly interesting are Euclidean and Hamming ones. In those spaces, efficient algorithms are often available for a specific task (like nearest neighbor search) and/or computing similarity or distances between vectors can be done faster than in the original “complex” space.

The idea behind embeddings is that smaller parts of objects are often sufficient to approximate distances or similarity, provided objects are partitioned in sufficiently random manner. In a number of interesting cases, this happens because of the phenomenon known as concentration of measure. An excellent state-of-the-art review of embeddings of general metrics as well as of special metrics such as edit distances is given in [Indyk, 2004].

Another research area where similar problems are considered is distributed representations that try to capture brain’s way of representing complex objects [Thorpe, 2003; Arbib, 2003]. In distributed representations objects of various complexity – from elementary to structured ones – are sought to be represented by patterns of activity over pools of "neurons", which can be thought of as "codevectors" (e.g., [Rachkovskij, Kussul, 2001]). It is believed that brain uses similar representations for an efficient recall and comparison of complex internal representations of real-world objects.

It was commonly thought that distributed representations are suitable only for "bag-of" representations [Feldman, 1989; Malsburg, 1986], however, the introduction of the so-called binding operations changed the situation [Plate, 2003]. In contrast to the non-distributed representations described above, here similarity of items is put into correspondence with the degree of correlation of their codevectors. Another property is the possibility of composing "reduced representations" of complex objects from subsets of codevectors of their parts as well as reconstructing full representations of parts from a reduced representation of the whole. Recursive construction of reduced representation results in a codevector representing the whole complex object. Different possibilities of combining components and reducing their representations give potential for constructing representations that reflect some necessary application-specific notion of similarity. One can use e.g. dot products or whatever for measuring similarity of codevectors – which are usually made of the same dimensionality even for items of different complexity.

For sequences, goals for embeddings and distributed representations are similar at least in the following: both try to find such a vector representation of sequences that preserves necessary application-specific similarity and provides fast calculation of similarity or difference in the target space. In this paper, we describe some ways of introducing order information into distributed representations, including those of binary sparse type; and their connection to traditional sequence vector representations and embeddings. We show that some non-distributed and distributed sequence-processing methods can be related through random embeddings (particularly, random projections) and can be viewed in a coherent way. We think that connecting sequence representation methods with embedding theory can help develop approach to their analysis and discover useful properties.

Vector Representation of Strings by q-grams

In this section, we mention some of non-distributed representations that can be characterized as bag-of representation approaches and that allow some approximation of edit distance.

Consider a string s and a sliding window of q symbols on it. For each possible window content (q-gram) the number of its occurrences in the string is recorded in a corresponding coordinate of the vector $v_q(s)$ (q-gram vector). In case of q=1 it is just a vector with elements equal to the number of times a respective letter was met in a sequence, equivalent to frequency vector in VSM. As it was noted in [Ukkonen, 1992], each edit operation changes at most q q-grams, so if the edit distance is at most L, then

$$||v_q(s)-v_q(t)|| \le qL \qquad (2)$$

This method discards order of seen q-grams. Nevertheless, this gives a way for approximating edit distance from below: $L \ge ||v_q(s)-v_q(t)||/q$ and thus can be used for filtering. Given a query string for which it is necessary to find the nearest one from a set of strings, too distant strings can be filtered out using Manhattan distance between q-gram vectors. Developed further, the idea of q-grams can be used to define such notions of similarity as "resemblance" and "containment" of strings [Broder, 1997].

Another interesting application of q-grams is solving gapped edit distance problem [Bar-Yossef et al, 2004]. To solve a k vs. m gapped edit distance problem is to be able to answer, given strings s and t, whether the edit

distance between them is less than k or it is greater than m. In the algorithm, each string is divided into non-intersecting regions of some length D. Then each q-gram is accompanied with a fingerprint that is equal to the number of region it starts within, constituting a set of pairs (γ, i), where $\gamma \in \Sigma^q$. Pairs are considered equal only if their q-grams are equal and they appear in the same region. So, identical q-grams, but starting far enough from each other, receive different fingerprints and are different. Then Hamming distance is measured between vectors corresponding to the sets of such pairs (γ, i). It turns out that it is possible to solve k vs. $(kn)^{2/3}$ gapped edit distance problem by measuring Hamming distance between characteristic vectors of sets of fingerprinted q-grams. Second step in [Bar-Yossef et al, 2004] is to use dimensionality-reducing technique in Hamming space from [Kushilevitz et al., 1998].

Distributed Sequence Coding

Let us consider some ideas of representing strings with distributed representations. Since symbols are considered non-similar, they are assigned independent random binary codevectors. Here we consider non-hierarchical version of order representation, that are designed for representation of short sequences – to represent long strings it may be necessary to build representations in a hierarchical manner.

Distributed representation of strings by unordered substrings. We will consider a distributed version of q-gram representation from the previous section. Let us allocate a random vector for each of the sequence substrings and sum up (or take disjunction of) them to form a codevector of the sequence. Sequences containing many identical elements are likely to receive close codes. This method takes information about order in the sequence into account only to an extent captured by chosen substrings.

This method is somewhat similar to the Random Indexing (RI) and Random Labeling (RL) methods used in semantic processing of texts, which are versions of vector space methodologies for producing distributed words' representations using co-occurrence data [Kanerva et al., 2000; Karlgren, Sahlgren, 2001]. There a word can be abstractly viewed as a set of contexts it is used in. Each context is a bag of words (either a text where the word was met (RI) or a word's neighborhood (RL)). Each element of a context (either a text or a word) is assigned a random vector (with small number of +1 and -1). Context representation is a sum of those random vectors corresponding to constituent words. Target word representation forms by adding those context vectors each time this word appears in a corpus. As we will see in the following sections, this method can be interpreted straightforwardly with embeddings.

The following methods are trying to merge information about position of an item within a sequence with the representation of the item itself by modifying item's representation in a position-dependent manner. In each of following schemes vectors can be made binary (sums could be substituted by disjunctions), making dot product (a common similarity measure) computation efficient.

Positional binding with permutations. Consider the so-called shift coding which exemplifies an attempt to preserve information about ordering in a string. Vectors are modified by circularly shifting (or otherwise permuting) item's vector by the number of bits that depends on the position of the item. Code of the whole sequence is formed by disjunction of codes of all constituent items.

Consider e.g. strings '*abc*', '*abd*', '*cba*'. Each symbol is represented by a random vector: $v(a)$, $v(b)$, $v(c)$, etc. Then sequence codevectors are formed in this way:

$$v(abc) = (v(a) >> 1) \vee (v(b) >> 2) \vee (v(b) >> 3)$$

$$v(bca) = (v(b) >> 1) \vee (v(c) >> 2) \vee (v(a) >> 3),$$

where $\vee$ is bit disjunction, $X>>y$ means codevector X shifted by y bits.

The intersection between obtained vectors for strings with no identical items in the same positions will be (in expectation) negligible, provided the random vectors are sufficiently sparse and strings of short enough length. Tthis coding scheme discards information about identical symbols in different positions. However, partial permutation or shift may provide a way to fix that.

Positional binding with codevector. Here codevectors for positions are generated additionally to those for substrings. To code a substring in a particular position, the codevector of the substring and the codevector of its position are bound. In the binary case, binding is done by bit-wise conjunction and is called

thinning [Rachkovskij, Kussul, 2001]. Other types of binding are also possible, both for binary codevectors [Rachkovskij, Kussul, 2001; Kanerva, 1996] and codevectors with continuous elements [Plate, 2003].

If we encode substring as continuous codevectors and positions as binary codevectors, binding by element-wise product is possible (see also Gayler's multiplicative binding for real-valued codevectors in [Plate, 2003]), which in this case can be also considered as thinning. Thinning does not remove similarity of identical elements in different positions, as the shift method does. Representations of positions may be correlated for nearby ordinal numbers. Codevectors of symbol-position bindings are then combined by bit-wise disjunction or addition.

Binding an element with its context. Another option is to bind item's vector with vectors of item's from its context. This may give way to build position-independent representation of items, where item's contribution depends only on its context and does not depend directly on the position in a sequence. As edit metrics is usually also position-independent in the sense of counting edit operations independently of the place they were applied; this may help to approximate them. Note an analogy with taking into account contexts by q-grams.

Embeddings and Representational Economy

In this section we discuss some results for embedding vector and sequence distances. Target spaces for embedding are usually vectors spaces like l_p (where $||x||_p=(\sum_{i=1,\dots,d} x_i^p)^{1/p}$), particularly interesting are Euclidean spaces (p=2), l_1 with Manhattan metrics ($||x||_1=\sum_{i=1,\dots,d}|x_i|$) or the Hamming space ($\rho(x,y)=\sum_{i=1,\dots,d}[x_i\neq y_i]$).

The seminal result that we will use is the Johnson-Lindenstrauss reduction lemma [Johnson, Lindenstrauss, 1984]. Let the elements of matrix $R_{d'\times d}$ be from $N(0,1)$ distribution. Let vectors $v'=Rv$. Then for every $\varepsilon>0$ and any two vectors $v_1,v_2 \in l^d_2$:

$$(1-\varepsilon)||v_1-v_2||_2 \le ||v'_1-v'_2||_2 \le (1+\varepsilon)||v_1-v_2||_2 \qquad (3)$$

holds with probability $\exp(\Omega(-d'/\varepsilon^2))$. In case of normal distribution of the vectors embedding into normed space is due to 2-stability of the normal distribution. There exist embeddings of norms for l_p using other p-stable distributions for $0<p\le 2$ (see [Cormode et al, 2002] for a brief overview). Thus it is possible to logarithmically reduce the input dimension while distorting mutual distance not too much. Note that instead of vector elements distributed according to normal distribution we can use either binary {-1,1} or ternary {-1,0,1} elements (with proper distribution) as proven in [Achlioptas, 2001].

Despite the progress in embedding "usual" metrics, embedding Levenshtein distance is a long-standing problem [Indyk, 2001], and a negative result was proved recently that any such algorithm would not have distortion smaller than 3/2 [Andoni et al., 2003]. However, it is possible to embed a relaxed version of edit distance (with block moves) to l_1 with approximately logarithmic distortion $\tilde{O}(log(n))$ by carefully selecting substrings, which add to the characteristic vector of the sequence [Cormode et al, 2000]. This result is particularly interesting because the exact calculation of this distance is NP-hard.

Connection between Distributed and Non-distributed Sequence Processing

In this subsection, we show how to interpret some distributed sequence coding methods with the help of embedding theory. We consider continuous case, so elements of the used vectors are from R and operations are usual summation and multiplication; as well as binary case, where operations are Boolean OR and AND.

Distributed representation of strings by unordered substrings According to it, a codevector $v(s)$ for a string $s=s_1,\dots,s_n$ is defined as $v(s)=\Sigma_{i=1,\dots,n} r(s_i)$, where $r(s)$ is a random codevector corresponding to an item s. The expression for $v(s)$ can be rewritten as

$$v(s) = \sum_{i=1,\dots,n} r(s_i) = \sum_{\sigma\in\Sigma}\sum_{i=1,\dots,n} I[s_i=\sigma] = \sum_{\sigma\in\Sigma} r(\sigma)\, n_s(\sigma), \qquad (4)$$

where $n_s(\sigma)$ is the number of times σ occurs in s.

Expression (3) is the same as projecting a bag-of-items vector $n_s^T(\sigma)=(n_1(\sigma),n_2(\sigma),\dots,n_{|\Sigma|}(\sigma))$ by multiplying it by a random matrix ($d\times|\Sigma|$) with columns $r(\sigma)$. Thus, this coding can be viewed as mapping from (e.g., VSM) space of dimension $|\Sigma|$ to R^d, where d can be made considerably lower then the number of all possible items $|\Sigma|$.

If both spaces are Euclidean, then, applying JL-lemma (1) and taking $r(s)$ from normal distribution (or from certain binary or ternary distributions, see [Achlioptas, 2001]) we obtain, that for a desired distortion $0<\varepsilon<1$, it is possible

to reduce dimension, while keeping $||v(s)-v(t)||$ within the range $(1\pm\varepsilon)||v_q(s)-v_q(t)||$ with high probability. This is what is done in RI (however, with sparse codevectors) appealing to "near orthogonality" of random vectors in high dimensions (but, in the same time, dimension can be considerably lower than the number of all contexts). And so using inequality (2), for the two strings within edit distance less then L the Euclidean distance between corresponding codevectors is no more then $(1+\varepsilon)qL$. This provides an upper bound for distance between vectors, and can be used for filtering purposes using representation vectors of low dimension instead of large (however, sparse) q-gram vectors.

There are experimental evidences that taking into account only information about presence of words in texts, it is possible to preserve similarity of texts to an extent sufficient for some applications [Grossman, Frieder, 1998].

Connection of the thinning coding to random sampling In this subsection we try to establish an analogy between thinning coding and some of the edit distance approximation approaches. Consider the i's element of output vector $v(s)$:

$$v_i(s) = \sum_{k=1,\dots,n} c_{ki}\, r_{i\,s[k]} = \sum_{k=1,\dots,n} c_{ki} \sum_{\sigma\in\Sigma} r_{i\sigma}\, I[s_k=\sigma] = \sum_{\sigma\in\Sigma}\sum_{k=1,\dots,n} r_{i\sigma}\, l_{\sigma k}\, c_{ki}. \quad (4)$$

Then we can define $V(s)=R\cdot L(s)\cdot C$, where columns of matrix R are random codevectors r_i assigned to the i'-th symbol of Σ and rows of C are position codevectors C_i.. Elements of the indicator matrix L are: $l_{\sigma k}=1$ if the k'-th substring (or symbol if $q=1$) of string s is σ and $l_{\sigma k}=0$ otherwise. That is, the element of the matrix shows where symbols from the alphabet (symbols or substrings) in the string are situated.

Consider the product $X=L(s)\cdot C$. If matrix C had contained only 1's in each of its cells, then this product would have just given columns of vector representation (e.g., q-gram) for the string. But, as C does contain zeros, substring in not all of the positions is counted into the vector. So, columns of the resulting matrix X would become an "incomplete" vector representations (q-gram representations, if Σ is all q-grams) of the input string s. Columns of matrix C act as binary masks marking positions in s, from where symbols sum up to a particular q-gram vector, e.g., the j's column of C masks s to obtain a vector with the i's element equal to $\sum_{k=1,\dots,n} l_{\sigma k} c_{ki}$. So, while earlier q-gram representations discarded order information in a sequence, here we introduced it with the help of position vectors.

Above, it was noted that position codevectors (rows of C) can be made so that nearby ones have small Hamming distance of each other and this distance grows as the distance between positions grows. Possible ways of constructing such codevectors from for ordinal numbers are considered in [Rachkovskij et al, 2005] and can also be implemented with a 2-state Markov chain with proper transition probabilities.

Different columns act as independent samplers. We note that similar approach has already led to even sublinear approximation of edit distance in [Batu et al, 2004]. Their approach is similar to ours in that the approximation is also achieved by randomly sampling input string and using the mentioned observation that strings within small edit distance have many substrings in nearby positions.

Consider now the effect of multiplying X by matrix R. Each i-th element of the output vector $v(s)$ is a dot product of the random codevector r_i by the corresponding "thinned" q-gram vector. We already saw such an operation (projection on a random direction) when established analogy between unordered distributed encoding of strings by assigning random codevectors to their substrings (q-grams). The difference is that there each of the q-grams was projected to all random directions, but here different thinned q-gram vectors are projected along different directions. However, if we look closer, we note that because of intersections between columns of C "common parts" of masked (thinned) q-gram vector may undergo projection to different directions. From expression (4):

$$v_i(s) = \sum_{\sigma\in\Sigma}\sum_{k=1,\dots,n} (c_{ki}.r_{i\sigma}) l_{\sigma k} = \sum_{k=1,\dots,n} R(C(k))\cdot l_k. \quad (4)$$

$R(C(k))$ is matrix R with only those rows left that correspond to those position codevectors that have 1s in position k. So, we see that each vector l_k undergoes projection to a respective random subspace, defined by those C_j that cover position k. The farther are identical symbols (or q-grams) from each other, the less common random directions they have, and so, the more distant are their projections.

Conclusions

We believe that enriching distributed representations with ideas and methods of analysis from embeddings can provide a more formal interpretation of distributed methods usually introduced in an ad-hoc manner. This may help infer the similarity type they approximate or find modifications that will allow them to approximate some, and help obtain bounds on the distortion of the proposed coding schemes, their complexity and limitations.

Here we have described approach to analysis of only few distributed schemes for coding sequences. We hope it could be extended to other schemes for encoding sequences mentioned above, as well as to schemes for distributed encoding of other types of data like numerical [Rachkovskij et al, 2005] or complex relational structures [Rachkovskij, 2004]. Those schemes include binding by context-dependent thinning and hierarchical representations [Rachkovskij, Kussul 2001].

Bibliography

[Achlioptas, 2001] D. Achlioptas, Database-friendly random projections. In Proceedings of PODS-01, pp. 274-281, 2001.

[Andoni et al., 2003] A. Andoni, M. Deza, A. Gupta, P. Indyk, S. Raskhodnikova, Lower Bounds for Embedding of Edit Distance into Normed Spaces. In Proc. of the 14th Symposium on Discrete Algorithms, 2003

[Arbib, 2003] M. Arbib, The Handbook of Brain Theory and Neural Networks. – Cambridge, MA: The MIT Press, 2003

[Bar-Yossef et al, 2004] Z. Bar-Yossef, T.S. Jayram, R. Krauthgamer, R. Kumar: Approximating Edit Distance Efficiently. FOCS, pp. 550-559, 2004

[Batu et al, 2004] T. Batu, F. Ergun, J. Kilian, A. Magen, S. Raskhodnikova, R. Rubinfeld, R. Sami. A sublinear algorithm for weakly approximating edit distance, In Proc. 36th STOC, 2004

[Broder, 1997] A. Z. Broder. On the resemblance and containment of texts. In Proceedings of Compression and Complexity of SEQUENCES, 1997

[Cormode et al, 2000] G. Cormode, M. Paterson, S. C. Sahinalp, U. Vishkin. Communication complexity of text exchange. In Proc. of the 11th ACM-SIAM Annual Symposium on Discrete Algorithms, pp. 197--206, San Francisco, CA, 2000

[Cormode et al, 2002] G. Cormode, P. Indyk, N. Koudas, S. Muthukrishnan. Fast mining of tabular data via approximate distance computations. In Proc. of the International Conference on Data Engineering, pp. 605–616, 2002

[Feldman, 1989] J. Feldman, Neural Representation of Conceptual Knowledge. In L. Nadel, L. Cooper, P. Culicover et al. Neural Connections, mental computation - London, England: The MIT Press, pp. 68-103, 1989

[Grossman, Frieder, 1998] D.A. Grossman, O. Frieder, Information Retrieval: Algorithms and Heuristics, Kluwer, 1998

[Indyk, 2001] P. Indyk, Algorithmic Aspects of Geometric Embeddings, FOCS, 2001

[Indyk, 2004] P. Indyk, Embedded Stringology, talk at the Symposium on Combinatorial Pattern Matching 2004. Available at http://theory.lcs.mit.edu/~indyk/cpm.ps

[Johnson, Lindenstrauss, 1984] W. Johnson, J. Lindenstrauss, Extensions of Lipschitz maps into a Hilbert space, Contemp. Math., 26 , pp. 189-206, 1984

[Kanerva et al, 2000] P. Kanerva, J. Kristofersson, A. Holst. Random indexing of text samples for latent semantic analysis. In Proc. of the 22nd Annual Conference of the Cognitive Science Society, p. 1036. Erlbaum, 2000

[Kushilevitz et al., 1998] E. Kushilevitz, R. Ostrovsky, Y. Rabani. Efficient Search for Approximate Nearest Neighbor in High Dimensional Spaces. STOC, pp. 614-623, 1998

[Levenshtein, 1965] V.I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals, Doklady Akademii Nauk SSSR, 163(4):845-848, 1965 (Russian)

[Lopresti, Tomkins, 1997] D. P. Lopresti, A. Tomkins, Block Edit Models for Approximate String Matching. Theoretical Computer Science, 181(1): 159-179, 1997

[Malsburg, 1986] C. von der Malsburg. Am I Thinking Assemblies? In Proc. of the Trieste Meeting on Brain Theory, pp.161-176, 1986

[Plate, 2003] T. Plate, Holographic Reduced Representation: Distributed Representation for Cognitive Structures. - Chicago: Center for the Study of Language and Information, 2003

[Rachkovskij et al, 2005] D.A. Rachkovskij, S.V. Slipchenko, E.M. Kussul , T.N. Baidyk, Sparse binary distributed encoding of scalar values, (to be published) 2005

[Rachkovskij, 2004] D.A. Rachkovskij, Some approaches to analogical mapping with structure sensitive distributed representations. Journal of Experimental and Theoretical Artificial Intelligence,16, №3, pp.125-145, 2004

[Rachkovskij, Kussul, 2001] D.A. Rachkovskij, E.M. Kussul, Binding and Normalization of Binary Sparse Distributed Representations by Context-Dependent Thinning. Neural Computation, 2, №13, pp.411-452, 2001

[Salton, 1989] G. Salton, G. Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer, Addison-Wesley, Reading, MA., 1989

[Thorpe, 2003] S. Thorpe, Localized Versus Distributed Representations. In Arbib M. The Handbook of Brain Theory and Neural Networks - Cambridge, MA: MIT Press, pp. 643-646, 2003

[Ukkonen, 1992] E. Ukkonen. Approximate string-matching with q-grams and maximal matches. Theoretical Computer Science, 92:191-211, 1992

Authors' Information

Artem M. Sokolov, Dmitri A. Rachkovskij – International Research and Training Center of Information Technologies and Systems; Pr. Acad. Glushkova, 40, Kiev, 03680, Ukraine; e-mails: sokolov@ukr.net, dar@infrm.kiev.ua