SEARCHING FOR NEAREST STRINGS WITH NEURAL-LIKE STRING EMBEDDING

Artem Sokolov

Abstract: We analyze an approach to a similarity preserving coding of symbol sequences based on neural distributed representations and show that it can be viewed as a metric embedding process.

Keywords sequence similarity, edit distance, metric embeddings, distributed representations, neural networks

ACM Classification Keywords: I.2.6 Connectionism and neural nets, E.m Miscellaneous

Introduction

Edit distance (Levenshtein distance) [Levenshtein, 1966] is used in a large number of research areas from genetics and web-search to anomaly detection in network traffic and voice recognition. Taking into account the contemporary data sequences' lengths (millions and billions of symbols) that have to be dealt with in the mentioned areas, the classic $O(n^2)$ edit distance calculation [Vintsyuk, 1968; Wagner, 1974] is not applicable in practice.

These circumstances gave birth to a branch of information theory concerned with the acceleration of edit distance calculation or its approximation (see survey [Navarro, 2001]). An exponential increase of the characteristic lengths of sequences, which are subject to comparison (the genome assembly, the need to compare data flows in information systems, etc.) urged interest to applications of the metric embedding theory (see survey [Indyk, 2004]). This theory is concerned with space mappings that simplify distance calculation [Indyk, 2001]. Levenshtein edit distance embedding to a vector space is known to be an actual open problem [Matoušek, 2002].

Independently, within the framework of the neural network paradigm of AI several approaches were proposed to the task of distributed representation and comparison of strings and other structured objects [Kussul, 1991; Rachkovskij, 2001]. Some approaches aimed at finding similarity of strings were presented in [Sokolov, 2005]. Here we develop one of them, namely, the approach based on the position-dependent thinning of vector representations, giving a theoretical grounding to the obtained scheme with the aid of probabilistic embedding of the edit metrics into the Manhattan space.

Task Description

We seek for a way to effectively calculate Levenshtein edit distance with the help of vector representations or, more specifically, by embedding edit metrics to a vector space. Our method belongs to the group of the so-called *q*-gram edit distance approximation methods (*q*-gram is a substring of length *q*), started by [Ukkonen, 1992]. We observed that the approach based on the distributed representations [Sokolov, 2005] resembles edit distance embedding or sketching methods [Cormode, 2000; Bar-Yossef, 2004; Batu, 2004] and therefore we attempted to combine both presenting the neural coding approach as an edit distance embedding into Manhattan space l_1. In order to show that the proposed method realizes one of the possible embedding definitions [Indyk, 2001], we will give the proofs of:

1) «upper bound», i.e. statements like ed(x,y) ≤ k_1 => P[d(v(x),v(y) ≤d_1] ≥ p_1 (1a)

2) «lower bound», .i.e. statements like ed(x,y) >k_2 => P[d(v(x),v(y) >d_2] ≥ p_2. (1b)

This definition (with appropriate values of the parameters envolved) covers embeddings that make discrimination between "near" and "far" strings (see subsection "LSH").

Mapping Description

For the two input strings *x,y* of length *n*, we independently and equiprobably select a sampling window of width *w* in both strings: *x[i,i+w-1]* and *y[i,i+w-1]*. Using fixed parameters of *q*-gram length q_1 and q_2, a *q*-gram vector $v_{w,q}$ is composed for each window for each $q=q_1,\ldots,q_2$ (vector of quantities of each *q*-gram appeared within a string). The obtained vectors corresponding to strings *x* and *y* are concatenated into vectors $v_q(x)$, $v_q(y)$. The Manhattan distance d^{Σ} between them would be the sum of Manhattan distances (i.e., $d_q(s,t) = \|v_q(s)-v_q(t)\|_1$) between *q*-gram vectors of the windows:

$$d^{\Sigma}(x[i,i+w-1],y[i,i+w-1]) = \sum_{q=q_1}^{q_2} d_q(x[i,i+w-1],[i,i+w-1]). \quad (2)$$

In the following, using the defined distance, we show that necessary embedding properties (1a) and (1b) hold. Some details and proofs omitted in this short paper will be presented elsewhere.

Lower Bound

De Bruijn graphs. For a string *x* and some parameter *q* de Bruijn graph [Bruijn, 1946] *B[x;q]*, $q \geq 3$ is a graph, whose vertices are all *(q-1)*-grams (*(q-1)*-spectrum) of the string *x*. An edge $a^1a^2\ldots a^q$ connects vertices labeled $a^1a^2\ldots a^{q-1}$ and $a^2a^3\ldots a^q$. Such graphs are widely used in genetics [Pevzner, 1989] and in cryptographic stream ciphers' analysis. Let a de Bruijn graph built using the union of *(q-1)*-spectra of two strings *x,y* be *B[x,y;q]*, and the path corresponding to a string *x* be π_x.

Let us consider possible local configurations of paths π_x *and* π_y on *B[x,y;q]*. Let us call the right and left branching points the vertices where the paths, correspondingly, diverge or converge. Let a "*half-loop*" be a subpath of either of the two paths stretching from a right branching point till the following left branching point in the path direction. The situation when there are no half-loops, i.e. in *B[x,y;q]* there is only one left or right branch point, or a left one and a right point following it, is called a "*fork*". In such a configuration, the left folk compulsorily contains as least one of the starting arcs of at least one of the paths, and a right folk contains terminating arcs of at least one of the ways. Let a "*shift*" be a special case of the fork, when there is a non-empty subpath $\pi_c, w\geq|\pi_c|\geq 0$, that $\pi_x=\pi'_x\pi_c$ and $\pi_y=\pi_c\pi'_y$, where π'_x, π'_y are some, possibly empty, subpaths.

A concept of "rotation" will be used to designate a way of obtaining identical spectra from different strings [Ukkonen, 1992, Pevzner, 1995]. A rotation is a situation when *(q-1)*-grams on the edges of a string are identical. It this case a corresponding path on the de Bruijn graph is a cycle, and starting from any of its vertices one can get different strings with the same spectrum.

Lemma 1 *Let* $x,y \in \Sigma^w$ *and there exist substrings* $x' \in x, y' \in y$, $x' \neq y'$, *such that x' is a rotation of y', then*

$$ed(x,y) \leq w + d_q(x,y)/2 - q + 1. \quad (3)$$

In the following we denote $\Delta q=q_2-q_1$, and $Q=(\Delta q+1)(\Delta q+2)$.

Our aim is to determine such a distance measure between two *w*-wide windows and such a threshold that strings with a distances less than this threshold would represent a shift and can be aligned in a fixed number of operations, by simply editing them at the beginning and the end of the window, namely, by eliminating forks at the edges of the windows. The usual *q*-gram distance (with a fixed *q*) cannot provide the desired result since for any *q* there can be found two strings *x,y*, where on the graph *B[x,y;q]* there will be half-loops.

Therefore we propose distance (2) and the following lemma states that with its aid it is possible to determine, whether the windows can be aligned with a small number of edit operations provided they do not include rotations of substrings.

Lemma 2 *Let* $x,y \in \Sigma^w$, *and x,y do not include substrings that are rotations of each other,* $q_2 > q_1 \geq 3$, $Q \leq 4(w - q_2+1)$,

$$d^{\Sigma}(x,y) < Q, \quad (4)$$

then $ed(x,y) < 2(\Delta q+1)$.

The next lemma unifies lemmas 1 and 2 by imposing conditions that allow applying both lemmas, correspondingly, in the case of presence of rotations and their absence.

Lemma 3 *Let* $q_2 > q_1 > ((w-3)^{1/2}+9)/4$, $w \geq 7$, $Q \leq 4(w - q_2+1)$, $Q<w$,

$$w-q_1 \leq \Delta q^2+5\Delta q/2 \quad \text{and} \quad d^{\Sigma}(x,y) < Q, \tag{5}$$

then $ed(x,y) < Q$.

We checked lemma 3 and 4 experimentally for those values of parameter w that still allowed for brute force string comparison. For a binary alphabet, all pairs of 2^w strings were compared for $w = 8,\ldots,17$ (experiment ran for 4 days). For a ternary alphabet all pairs of 3^w strings were compared for $w = 8,\ldots,10$ (2 days). None of the experiments has found a pair of strings violating the lemma.

Let there be two types of pairs of windows $x[i,i+w-1],y[i,i+w-1]$: “good” and “bad” – correspondingly those for which condition (4) holds or not. The next lemma says that it is possible to simultaneously align successive “good” windows.

Lemma 4 *Let conditions of lemma 3 hold,* $w=7,\ldots,n$, *if for all* $i=1,\ldots,n-w+1$ *it holds that* $d^{\Sigma}(x[i,i+w-1],y[i,i+w-1])<Q$, *then* $ed(x,y)<2Q$.

The next lemma defines the minimal possible distance between two “good” pairs of windows, with “bad” pair between them. Denoting $t=w-q_2-\Delta q$ we show that the maximum distance between two “good” pairs of windows without a possibility to contain a “bad” one between them is $t+2$.

Lemma 5 *Let* $x,y \in \Sigma^m$, $t > 2$ *and the conditions of lemma 3 hold, if for some* $i,j=1,\ldots,m-w+1$, $j>i$, $j-i < t+2$

$$\begin{gathered} d^{\Sigma}(x[i,i+w-1],y[i,i+w-1]) < Q, \\ d^{\Sigma}(x[j,j+w-1],y[j,j+w-1]) < Q, \end{gathered} \tag{6}$$

then for all $i'=i,..,j$ $d^{\Sigma}(x[i',i'+w-1],y[i',i'+w-1]) < Q$.

Let N be the number of “bad” windows. Let us find the upper limit (lemma 6) on the edit cost for all possible arrangements of N windows, using the following string edit algorithm. Assume we have aligned strings up to position $j-1$. If all consecutive pairs of windows, beginning from position j and to $j+r$, are “good”, then we align them with not more than $2Q$ operations, using the result of lemma 4, and continue from position $j+r+w-1$. If the next pair of windows in position j is bad, we use one edit operation to replace symbol $x[j]$ with symbol $y[j]$, thus aligning one symbol and continuing to the next pair of windows in position $j+1$.

Lemma 6 *Let conditions of lemma 3 hold. Let* $T=\lfloor(n-1)/w\rfloor$. *The cost of aligning* strings *y and x with the help of the above algorithm is upper bounded with*

$$\max(Q+N, (2Q-1)\min(T,N)+\min(N,n-1-(w-1)\min(T,N))+2Q) \leq 2Q(\lceil N/t\rceil+1) \tag{7}$$

Finally, for the independent and equiprobable window sampling we get from (7) the following lemma that specifies property (1b):

Lemma 7 *For* $x,y \in \Sigma^n$ *and holding conditions of lemma 3, if* $ed(x,y) > k_2$,, *then*

$$P[d^{\Sigma}(x,y) \geq Q] > (t\,k_2/2Q-2)\,/\,(n-w+1). \tag{8}$$

Nearest Neighbor Search

In this part we consider a possible procedure for the nearest string search (NNS) with the help of edit distance approximation described above. Let $P=\{p_1,\ldots,p_P|p_i \in P\}$ be a collection of strings and p_0 be the input probe string, to which it is necessary to find the nearest one (by the edit distance) from P. Searching for the exact nearest neighbor is often a laborious task. Namely, for large dimensionalities of the input space (in our case it is $d=|\Sigma|^n$) the existing NNS algorithms are reduced to the linear search on P. On the other hand, the “approximately” nearest neighbor is often sufficient in applications and it is often much easier to find it.

First we transform vectors v_q into hash-values distributed around $||v_q||$ with the help of a p-stable distribution, and a modified scheme from [Datar, 2004] (see subsection "Modification"). Then we apply the well-known scheme of locality-sensitive hashing (LSH) [Indyk, 1998].

LSH. Let us describe the original [Indyk, 1998] LSH scheme applied to strings with the classic edit metrics. Define a ball with radius r containing points distanced from its center not farther then r : $S(t,r)=\{q \mid ed(q,s) \leq r\}$.

Definition of locality-sensitive functions. A family of hash-functions $H=\{h: \Sigma \to X\}$ is called (r_1,r_2,p_1,p_2)-sensitive, if for any $x,y \in \Sigma^n$ and any independently and equiprobably chosen $h \in H$ holds the following:

$$t \in S(s,r_1) \Rightarrow P[h(t)=h(s)] \geq p_1 \quad \text{and} \quad t \notin S(s,r_2) \Rightarrow P[h(t)=h(s)] \leq p_2, \\ r_1<r_2 \quad \text{and} \quad p_1<p_2 \tag{9}$$

Compose random hash-vectors $g_j=(h_1,\ldots,h_K)$, $j=1,\ldots,L$ from functions h. Additionally, we create cells where we put a string $p_i \in P$ based on the value of the hash-vector $g(p_i)$: a string p_i is put into a cell with an identifier equal to the hash-vector value. The aim is to get high collision probability between nearby strings, and low probability between distant ones. Then, applying the same hash to the probe we check whether it equals one of the previously stored hashes of vectors from P: for probe p_0 we calculate all hash-vectors $g_i(p_0)$, $i=1,\ldots,L$ and examine corresponding cells. If some cell contains a string $p^*_i \in S(p,r_2)$, the algorithm returns YES and p^*_i and NO otherwise (thus representing a solution to the so called (r_1,r_2)-PLEB task [Indyk, 1998]). The algorithm terminates after checking $2l$ cells. For such a procedure, the following theorem holds:

Theorem 1 [Indyk, 1998] *Let H be a (r_1,r_2,p_1,p_2)-sensitive family of functions, K=-ln$|P|$/ln(p_2), $L=|P|^\rho$, where ρ=ln(p_1/p_2). Then the above algorithm solves (r_1,r_2)-PLEB task and takes $O(|\Sigma|^q|P|+|P|^{1+\rho})$ space, $O(|P|^\rho)$ distance calculations, and $O(|P|^\rho K)$ calculations of hash functions.*

LSH with a 1-stable distribution. In [Datar, 2004], it is proposed to use a particular property of stable distributions, that linear combinations of theirs random values ϕ_i are distributed as one such random variable multiplied by the norm of linear combination's coefficients. Due to linearity of scalar product, (v_1,ϕ) - (v_2,ϕ): $||v_1\text{-}v_2||_{lp}\phi$. Hash-functions are defined as:

$$h(v)=\lfloor((v_q,\boldsymbol{\phi})+b)/r\rfloor, \tag{10}$$

where b is an equiprobably distributed random variable on $[0,r]$, $\boldsymbol{\phi}$ is a vector with elements taken from Cauchy distribution. If one divides a real axis into equal intervals, then, intuitively, vectors with the similar norm will likely fall into the same interval. It is possible to show [Datar, 2004] that for two fixed vectors the hash-function (10) is

$$p(c) = \int_0^r \frac{1}{c} f(c)(1-\frac{t}{c})dt = \frac{1}{\pi}\left(2\arctan\left(\frac{r}{c}\right) - \frac{c}{r}\ln\left(1+\left(\frac{r}{c}\right)^2\right)\right) \tag{11}$$

where $f(.)$ is the probability density function of the absolute value of ϕ, and $c=||v_1\text{-}v_2||$ is the distance between the hashed vectors. As $p(c)$ is a monotonically decreasing function, the family of such functions is locality-sensitive (see (9)). So, hash-functions (10) can be used in the LSH scheme.

Modification. We will use the hash-functions (10) in a different way from that described above, to pursue the analogy to the distributed approaches [Sokolov, 2005]. Instead of multiplying a fixed vector v_q by a number of random vectors ϕ to form hash-vectors g_j, we take K random vectors $v^i_q(s)$ obtained by random and independent sampling with a window of width w from string s (see "Mapping description"). For each of them, we generate a separate random vector ϕ_i whose elements are taken from the Cauchy distribution. Let us designate the scalar product of these two vectors $h'_i=(v^i_q(s),\phi_i)$ and fix the hash-functions of the form $h'(v)=\lfloor(h'_i+b)/r\rfloor$. Taking into account lemmas 1 and 7, the following lemma holds indicating that the family of such functions is also locality-sensitive.

Lemma 8 Let *function p(.) be defined as in (11). For the collision probability of hash functions h' it holds*

$$P[h'(x)=h'(y) \mid ed(x,y) \leq k_1] \leq p(2k_1(\Delta q+1))(1\text{-}k_1w/(n\text{-}w+1)), \\ P[h'(x)=h'(y) \mid ed(x,y) \geq k_2] \leq 1\text{-}\ (tk_2/2Q\text{-}2)(1\text{-}p(Q)). \tag{12}$$

With this method of hash-function formation, we get output vectors without matrix multiplying of intermediate *q*-gram representations by random vectors ϕ, thus obtaining a scheme consistent with the neural network approach [Sokolov, 2005].

In the original setting parameter *r* in (10) can be chosen, e.g., to minimize p_1/p_2 (see [Datar, 2004]) and to speed up the NNS procedure (theorem 1). However, here we will use *r* to fulfill requirements (9) on r_1, r_2 и $p_1, p_2,$ that, together with (12), leads to the conclusion that we should have

$$k_2>2Q\left(2+\frac{n-w+1}{t\left(1-p(d_2)\right)}\left(1-p(d_1)+\frac{wk_1p(d_1)}{n-w+1}\right)\right). \tag{13}$$

This should be at most $\Theta(n)$ for the property (1b) not to become trivial. We can achieve this by letting *r* be some function of the string length *n,* e.g. $r=n^{\mu}$. We also set $w=n^{\gamma}$. Taking into account asymptotic behavior of $t=\Theta(w)$, $\Delta q=\Theta(w^{1/2})$, $Q=\Theta(w)$, analysis shows, that the optimal values for the parameters are $\mu = \gamma = 2/3$. And from (13), the lower bound on the k_2 growth rate is

$$k_2 = \Omega(k_1 n^{2/3}\ln n) \tag{14}$$

Parameter ρ (see theorem 1) determines search efficiency and resource requirements for the described LSH scheme. If , for $\varepsilon > 1$, k_2 is chosen as follows:

$$k_2= 2Q\left(2+\frac{\varepsilon(n-w+1)}{t\left(1-p(d_2)\right)}\left(1-p(d_1)+\frac{wk_1p(d_1)}{n-w+1}\right)\right),$$

it can be shown, using the same method as for locality-sensitive family of functions for the Hamming distance [Indyk, 1998], that $\rho = O(1/(1+\varepsilon))$.

Ternarization. It is rather attractive to have either binary or ternary vectors at the output, because they are more beneficial than integer-valued ones because of a more efficient implementation. Moreover, (sparse) binary or ternary vectors are widely used in distributed processing models [Rachkovskij, 2001]. Hash-function (10) will take ternary values $\{-1,0,1\}$ if $1 < h(v) < 2$ and so $-r < (\phi,v) < r$. Integrating $(\pi(1+x^2))^{-1}$ with limits from $-r/||v||$ to $r/||v||$ we get for the percentage of ternary elements in the output: $2\arctan(r/||v||)/\pi$. Density of zero elements is an important parameter in distributed representations, and for ternary vectors it is given by $\arctan(r/||v||-||v|| \ln(1+(r/||v||)^2)/2r)/\pi$ and is increasing with the growth of $r=n^{2/3}$.

Conclusion

We analyzed the concept of the distributed representations of sequences [Sokolov, 2005] from the point of view of metric embeddings, presented a new *q*-gram approximation method of the edit distance, and proved the possibility of constructing locality-sensitive functions. Thus we showed that the distributed representations used for the comparison of sequential data in the neural network paradigm could be justified with the aid of the methods from the embedding theory. This approach can also be considered as the substantiation of the Broder approach [Broder, 1995] who takes for the document similarity measure the degree of the coincidence of the sets of their *q*-grams and also other bag-of-grams methods. We also gave conditions for obtaining binary and ternary vectors at the output that can be useful for a unified approach to representation and processing of various data types and modalities [Rachkovskij, 2001].

A prospective direction of further work may be to check if lemma 3 can could be strengthened to guarantee $ed(x,y) < 2(\Delta q+1)$ with modified conditions on q_1 and Δq. If the available preliminary experimental indications of this are proved, it would lead to a considerable improvement of the lower bound (14) in the modified LSH scheme and in a deterministic variant of the mapping.

Bibliography

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

[Batu, 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, 1998] A. Broder. On the resemblance and containment of documents. In *SEQS: Sequences '97*, 1998.

[Bruijn, 1946] N. G. de Bruijn. A combinatorial problem. In *Koninklijke Nederlandsche Akademie van Wetenschappen*, volume 49, 1946.

[Cormode, 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

[Datar, 2004] M. Datar, N. Immorlica, P. Indyk, V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *20-th annual symposium on Computational geometry*, pages 253–262, Brooklyn, New York, USA, 2004.

[Indyk, 1998] P. Indyk, R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proc. of 30th STOC*, pp. 604–613, 1998.

[Indyk, 2001] P. Indyk. Algorithmic aspects of geometric embeddings. In *FOCS*, 2001.

[Indyk, 2004] P. Indyk. Embedded stringology. Talk at *15-th Annual Combinatorial Pattern Matching Symposium*, July 2004.

[Kussul, 1991] Kussul, E. M., & Rachkovskij, D. A. (1991). Multilevel assembly neural architecture and processing of sequences. In A .V. Holden & V. I. Kryukov (Eds.), Neurocomputers and Attention: Vol. II. Connectionism and neurocomputers (pp. 577-590). Manchester and New York: Manchester University Press.

[Levenshtein, 1966] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics - Doklady*, 10(8):707–710, February 1966.

[Matoušek, 2002] Open problems. In J. Matoušek, editor, *Workshop on Discrete Metric Spaces and their Algorithmic Applications*, Haifa, March 2002.

[Navarro, 2001] G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, 2001.

[Pevzner, 1989] P. A. Pevzner,P. L-tuple DNA sequencing: computer analysis. *J. Biomol. Struct. Dyn.*, 7, 63–73,1989

[Pevzner, 1995] P. A. Pevzner. DNA physical mapping and alternating eulerian cycles in colored graphs. *Algorithmica*, 13(1/2):77–105, 1995.

[Rachkovskij, 2001] D. A. Rachkovskij, E. M. Kussul. Binding and Normalization of Binary Sparse Distributed Representations by Context-Dependent Thinning, *Neural Comp*. 13: 411-452, 2001.

[Shamir, 2004] R. Shamir. Lecture notes in Analysis of Gene Expression Data, DNA chips and Gene Networks: Sequencing by hybridization. www.cs.tau.ac.il/~rshamir/ge/04/scribes/lec02.pdf, 2004.

[Sokolov, 2005] A. Sokolov, D. Rachkovskij. Some approaches to distributed encoding of sequences. In *Proc. of XI-th International Conference Knowledge-Dialogue-Solution*, volume 2, pp. 522–528, Varna, Bulgaria, June 2005.

[Ukkonen, 1992] E. Ukkonen. Approximate string-matching with q-grams and maximal matches. *Theor. Comput. Sci.*, 92(1):191–211, 1992.

[Vintsyuk, 1968] T. K. Vintsyuk. Speech discrimination by dynamic programming. *Kibernetika (Cybernetics)*, (4):81–88, 1968.

[Wagner, 1974] R. A. Wagner, M. J. Fischer. The string-to-string correction problem. *J. of the ACM*, 21(1):158–173, 1974.

Authors' Information

Artem M. Sokolov – International Research and Training Center of Information Technologies and Systems; Pr. Acad. Glushkova, 40, Kyiv, 03680, Ukraine; e-mail: sokolov (at) ukr.net