Algorithm for Computing the Similarity between Binary Vectors

Vladimir B. Balakirsky
"Confident" (Russia)
EIDMA (The Netherlands)
Email : v.b.balakirsky@tue.nl

Abstract — **The algorithm for computing the length of a common subsequence of two binary sequences of length n, which requires a variable number of computations depending on input data, is presented. The basis of the algorithm can be effectively used to obtain the probability distribution of the similarity between random vectors. This point is demonstrated for the case when one of vectors is chosen at random and another one is the alternating vector.**

Let $S(\mathbf{u},\mathbf{v})$ denote the length of the longest sequence occurring as a common subsequence between two given binary sequences, $\mathbf{u}$ and $\mathbf{v}$, of the same length n. The parameter $S(\mathbf{u},\mathbf{v})$ is called the deletion similarity between $\mathbf{u}$ and $\mathbf{v}$ [1], and $n - S(\mathbf{u},\mathbf{v})$ is called the Levenshtein distance between $\mathbf{u}$ and $\mathbf{v}$. This parameter is important for designing insertions–deletions correcting codes. It is also relevant to other applications in cryptology and data compression.

Definition 1. *The similarity $S(\mathbf{u}_i,\mathbf{v}_j)$ between two binary vectors, $\mathbf{u}_i = (u_1,\ldots, u_i)$ and $\mathbf{v}_j = (v_1,\ldots,v_j)$, is the largest integer s such that there exist two vectors, $(i_1,\ldots,i_s)$ and $(j_1,\ldots,j_s)$ with $1 \le i_1 < i_2 < \ldots < i_s \le i$ and $1 \le j_1 < j_2 < \ldots < j_s \le j$, such that $u_{i_d} = v_{j_d}$ for all $d = 1,\ldots,s$. The similarity between the all–zero vector and the all–one vector is defined as 0.*

We will also use the notations $S_{i,j} = S(\mathbf{u}_i,\mathbf{v}_j)$ and $\mathbf{u} = \mathbf{u}_n$, $\mathbf{v} = \mathbf{v}_n$.

The algorithms that allow us to compute $S(\mathbf{u},\mathbf{v})$ with the complexity $O(n^2)$ are known [2], [3]. We address the problem of simplification of these algorithms.

Let us denote $[n] \triangleq \{1,\ldots,n\}$. For $i = j = 0$ and for all $i,j \in [n]$ with $u_i = v_j$ introduce $a_{i,j}, b_{i,j} \ge 1$ as the smallest integers such that

$$u_{i+1} = v_{j+b_{i,j}}, \quad v_{j+1} = u_{i+a_{i,j}},$$

where we assume that $u_{i+1} = v_{n+1}$ and $v_{j+1} = u_{n+1}$.

N0. Set $s = 0$ and $\mathcal{N}_0 = \{(0,0)\}$.

N1. Form the sets

$$\mathcal{N}'_{s+1} = \bigcup_{(i,j)\in\mathcal{N}_s} \Big\{ (i+1, j+b_{i,j}),\ (i+a_{i,j}, j+1) \Big\}$$

and $\mathcal{N}_{s+1} \subseteq \mathcal{N}'_{s+1}$ in such a way that the pair $(i',j') \in \mathcal{N}'_{s+1}$ does not belong to the set $\mathcal{N}_{s+1}$ if and only if one of the following two conditions are satisfied : (a) $i' > n$ or $j' > n$; (b) there exists a pair $(i^*,j^*) \in \mathcal{N}'_{s+1}\backslash\{(i',j')\}$ either with $i^* < i', j^* = j'$ or with $j^* < j', i^* = i'$.

N2. If $\mathcal{N}_{s+1} \ne \emptyset$, then increase s by 1 and go to N1.

N3. Output s.

The N0–N3 algorithm is illustrated in Table 1.

A "direct algorithm" for finding the similarity between $\mathbf{u}$ and $\mathbf{v}$ can be presented as sequential computations of the elements of an $n \times n$ matrix $\mathbf{S}$ (for example, row by row) and output the value of $S_{n,n}$ using the following rules :

$$\begin{aligned} u_i \ne v_j &\implies S_{i,j} = S^*_{i,j}, \\ u_i = v_j &\implies S_{i,j} = \max\Big\{ S^*_{i,j},\, S_{i-1,j-1} + 1 \Big\}, \end{aligned}$$

Table 1: The entries of the sets constructed by the N0–N3 algorithm for the pair of vectors $(\mathbf{u},\mathbf{v}) = (001010, 110001)$; $S(\mathbf{u},\mathbf{v}) = 4$ and $C(\mathbf{u},\mathbf{v}) = 10$.

s	$\mathcal{N}_s$	$\mathcal{N}'_{s+1}$
0	(0,0)	(1,3), (3,1)
1	(1,3)	(2,4)
	(3,1)	(4,3), (5,2)
2	(2,4)	(3,6), (4,5)
	(4,3)	(5,6), (6,4)
	(5,2)	(6,3)
3	(3,6)	(4,7)
	(4,5)	(5,6)
	(6,3)	(7,4)
4	(5,6)	(6,7)

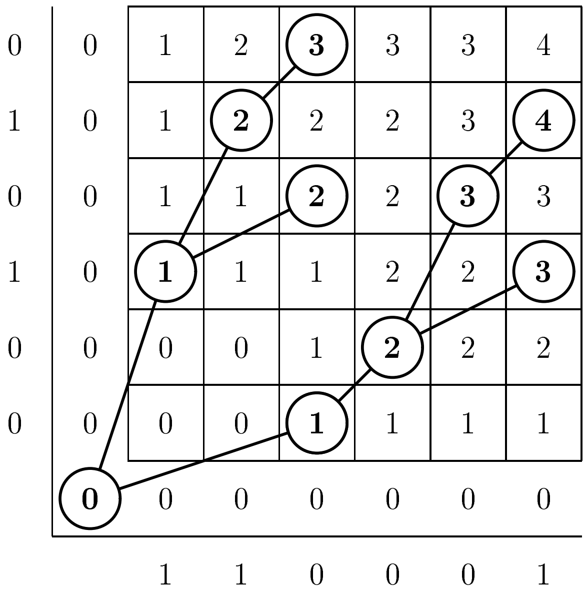

Figure 1: Illustration of the algorithm for computing the similarity between two vectors, $\mathbf{u} = 001010$ and $\mathbf{v} = 110001$, and corresponding similarity tree.

where

$$S^*_{i,j} \triangleq \max\Big\{ S_{i-1,j},\ S_{i,j-1} \Big\}$$

and

$$\text{either } i' = 0 \text{ or } j' = 0 \implies S_{i',j'} \triangleq 0.$$

This algorithm is illustrated in Figure 1.

For all $i, j \in [n]$, let us denote

$$T_{i,j} \triangleq \begin{cases} S_{i,j}, & \text{if } S_{i,j} > S^*_{i,j}, \\ 0, & \text{otherwise.} \end{cases}$$

Furthermore, let $T_{0,0} \triangleq 0$. Introduce an oriented graph whose nodes are identified by the pair $(0,0)$ and by all pairs $(i,j) \in [n] \times [n]$ such that $T_{i,j} = S_{i,j}$. Define the edges of the graph as follows : there is an edge connecting the node (i,j) with the node (i',j') if and only if

$$T_{i',j'} = T_{i,j} + 1,\ i' > i,\ j' > j.$$

One can easily check that the graph above is a rooted oriented ordered binary tree, which we call *the similarity tree* and illustrate the construction in Figure 1. The difference between the N0–N3 algorithm and "the direct algorithm" is that we immediately build a tree and form $S(\mathbf{u},\mathbf{v})$ as the largest value of $T_{i,j}$, where the maximum is taken on all pairs $(i,j) \in [n] \times [n]$.

Theorem 1. *For any pair of vectors* $(\mathbf{u},\mathbf{v})$, *the* N0–N3 *algorithm outputs* $s = S(\mathbf{u},\mathbf{v})$.

Notice that $a_{i,j}$ and $b_{i,j}$ can be easily found after $\mathbf{u}$ and $\mathbf{v}$ are represented as results of concatenations of the 0–packets and the 1–packets when the lengths of packets are computed in advance and stored in the memory. To "refine" the set $\mathcal{N}'_{s+1}$ at step N1(b) we also need the number of computations, which is linear in $|\mathcal{N}'_{s+1}|$. Namely, let $\mathbf{I} = (I_0, \ldots, I_n)$ and $\mathbf{J} = (J_0, \ldots, J_n)$ be two auxiliary arrays. Suppose that we have inspected some number of entries of the set $\mathcal{N}'_{s+1}$, formed the set $\mathcal{N}_{s+1}$ of cardinality M, and numbered its entries by $(i_1, j_1), \ldots, (i_M, j_M)$. Furthermore, let the arrays $\mathbf{I}$ and $\mathbf{J}$ be filled in such a way that

$$\begin{cases} I_{i_m} = J_{j_m} = m, & m = 1, \ldots, M; \\ i \notin \{i_1, \ldots, i_M\} & \implies\ I_i = 0; \\ j \notin \{j_1, \ldots, j_M\} & \implies\ J_j = 0. \end{cases}$$

Then the inspection of the next entry $(i',j') \in \mathcal{N}'_{s+1}$ proceeds as follows.

1. If $I_{i'} = J_{j'} = 0$, then set $(i_{M+1}, j_{M+1}) =$

(i', j') and $I_{i'} = J_{j'} = M + 1$. Increase M by 1.

2. If $I_{i'} = 0$ and $J_{j'} = r > 0$, then compare i' with i_r. If $i' > i_r$, then ignore the pair (i', j'). Otherwise, replace (i_r, j_r) with (i', j') by setting $i_r = i'$.

3. If $J_{j'} = 0$ and $I_{i'} = r > 0$, then compare j' with j_r. If $j' > j_r$, then ignore the pair (i', j'). Otherwise, replace (i_r, j_r) with (i', j') by setting $j_r = j'$.

The number of computations required by the N0–N3 algorithm can be measured by the sum of cardinalities of the sets $\mathcal{N}_s(\mathbf{u}, \mathbf{v}) = \mathcal{N}_s$, $s = 0, \ldots, S(\mathbf{u}, \mathbf{v})$, formed for a given pair of vectors $(\mathbf{u}, \mathbf{v})$. Denote

$$C(\mathbf{u}, \mathbf{v}) \ \stackrel{\triangle}{=} \ \sum_{s=0}^{S(\mathbf{u},\mathbf{v})} |\, \mathcal{N}_s(\mathbf{u}, \mathbf{v})\,|.$$

In Table 2 we show the average and the maximum number of computations in accordance with the function C :

$$\overline{C}_n \ \stackrel{\triangle}{=} \ 2^{-2n} \sum_{\mathbf{u},\mathbf{v}} C(\mathbf{u}, \mathbf{v}),$$

$$C_{\max,n} \ \stackrel{\triangle}{=} \ \max_{\mathbf{u},\mathbf{v}} C(\mathbf{u}, \mathbf{v}),$$

as well as the values of the average similarity $\overline{S}_n$ and the entropy H_n for the probability distribution $(P_n(0), \ldots, P_n(n))$ having components

$$P_n(s) \stackrel{\triangle}{=} 2^{-2n} \Big| \Big\{ \, (\mathbf{u}, \mathbf{v}) : \ S(\mathbf{u}, \mathbf{v}) = s \, \Big\} \Big|.$$

Theorem 2. *If $n + 1$ is divisible by 3, then the maximum number of computations can be expressed as*

$$C_{\max,n} = C(\tilde{\mathbf{u}}, \tilde{\mathbf{v}}) = \frac{(n+1)(2n+5)}{9} \approx \frac{2n^2}{9},$$

where

$$\tilde{\mathbf{u}} \ \stackrel{\triangle}{=} \ 1^{\ell}(01)^{(n-\ell-1)/2}0,$$

$$\tilde{\mathbf{v}} \ \stackrel{\triangle}{=} \ 0^{\ell-1}(10)^{(n-\ell+1)/2},$$

and $\ell \stackrel{\triangle}{=} (n+1)/3$.

Table 2: Some parameters of the probability distribution of the similarity between two random binary vectors of length n.

n	$\overline{S}_n/n$	$H_n/\log n$	$\overline{C}_n/n^2$	$C_{\max,n}/n^2$
4	0.631	0.813	0.269	0.437
5	0.649	0.752	0.224	0.400
6	0.663	0.711	0.197	0.361
7	0.674	0.684	0.178	0.347
8	0.684	0.662	0.165	0.328
9	0.691	0.646	0.155	0.321
10	0.698	0.632	0.147	0.310
11	0.704	0.621	0.140	0.305
12	0.708	0.612	0.135	0.298
13	0.713	0.604	0.130	0.295
14	0.717	0.596	0.127	0.290
15	0.720	0.590	0.123	0.288
16	0.724	0.585	0.120	0.285

Finding of the analitical expression for the probability distribution of the similarity seems to be a difficult combinatorial problem. One of approaches to attack this problem is the finding of the probability distribution of the similarity between a uniformly distributed random vector $\mathbf{u}$ and a vector $\mathbf{v}$, which is fixed in a regular way. This distribution highly depends on $\mathbf{v}$, as it is illustrated in Table 3 and Figure 2 where we show the values of the average similarity $\overline{S}_n(\mathbf{v})$ and the entropy $H_n(\mathbf{v})$ for the probability distribution $(P_n(0|\mathbf{v}), \ldots, P_n(n|\mathbf{v}))$ having components

$$P_n(s|\mathbf{v}) \stackrel{\triangle}{=} 2^{-n} \Big| \Big\{ \, \mathbf{u} : \ S(\mathbf{u}, \mathbf{v}) = s \, \Big\} \Big|.$$

Theorem 3. *Suppose that n is even and denote $\mathbf{v}_2 \stackrel{\triangle}{=} (01)^{n/2}$. The similarity between $\mathbf{v}_2$ and any binary vector $\mathbf{u}$ of length n having $k(\mathbf{u})$ packets can be expressed as*

$$S(\mathbf{u}, \mathbf{v}_2) = \begin{cases} n/2 + \lceil (k(\mathbf{u}) - 1)/2 \rceil, & \text{if } u_1 = 0, \\ n/2 + \lfloor (k(\mathbf{u}) - 1)/2 \rfloor, & \text{if } u_1 = 1, \end{cases}$$

where

$$k(\mathbf{u}) \stackrel{\triangle}{=} 1 + \Big| \Big\{ \, i \in \{2, \ldots, n\} : \ u_i \neq u_{i-1} \, \Big\} \Big|.$$

Table 3: Some parameters of the probability distributions of the similarity between random vectors and fixed vectors $\mathbf{v}$ of length n.

$n=16$			$n=8$		
$\mathbf{v}$	$\overline{S}_n(\mathbf{v})/n$	$H_n(\mathbf{v})/\log n$	$\mathbf{v}$	$\overline{S}_n(\mathbf{v})/n$	$H_n(\mathbf{v})/\log n$
$(01)^8$	0.734	0.522	$(01)^4$	0.718	0.543
$(0^21^2)^4$	0.749	0.502	$(0^21^2)^2$	0.714	0.542
$(0^41^4)^2$	0.708	0.519	0^41^4	0.637	0.601
0^81^8	0.621	0.587	0^8	0.5	0.848
0^{16}	0.5	0.761			

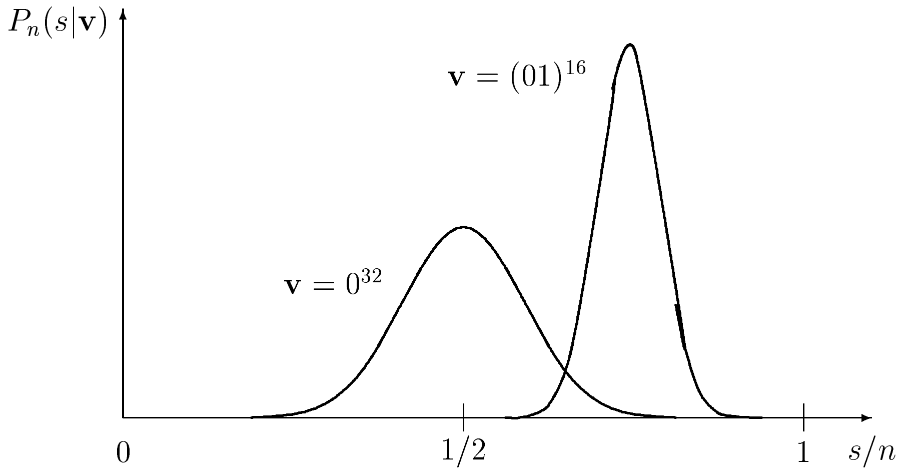

Figure 2: The probability distributions of the similarity between random vectors of length $n = 32$ and the vectors $\mathbf{v} = 0^{32}, (01)^{16}$.

Corollary.

$$P_n(s|\mathbf{v}_2) = \binom{n+1}{2s-n} 2^{-n}, \quad s = n/2, \ldots, n$$

and

$$P_n(s|\mathbf{v}_2) = 0, \quad \textit{for all } s < n/2.$$

Thus,

$$\overline{S}_n(\mathbf{v}_2) = \frac{3n-1}{4}.$$

References

[1] V. I. Levenshtein, "Binary codes capable to correct deletions, insertions, and reversals", *Doklady Akademii Nauk SSSR*, vol. 163, pp. 845–848, 1965 (in Russian). English translation : *J. Soviet Phys.–Doklady*, vol. 10, pp. 707–710, 1966.

[2] R. A. Wagner and M. J. Fischer, "The string–to–string correction problem", *Journal of the ACM*, vol. 21, no. 1, pp. 168–173, 1974.

[3] W. J. Masek and M. S. Paterson, "How to compute string–edit distance quickly", in : *Time Warps, String Edits, and Macromolecules : The Theory and Practice of Sequence Comparison*, D. Sankoff and J. B. Kruskal (eds.), Addison–Wesley, pp. 337–349, 1983.