Transition-Sensitive Distances

Kaoru Yoshida

Sony Computer Science Laboratories, Inc.
3-14-13 Higashigotanda, Shinagawa-ku, Tokyo, 141-0022 Japan
kaoru@csl.sony.co.jp

Abstract. In information retrieval and classification, the relevance of the obtained result and the efficiency of the computational process are strongly influenced by the distance measure used for data comparison. Conventional distance measures, including Hamming distance (HD) and Levenshtein distance (LD), count merely the number of mismatches (or modifications). Given a query, samples mapped at the same distance have the same number of mismatches, but the distribution of the mismatches might be different, either disperse or blocked, so that other measures must be cascaded for further differentiation of the samples. Here we present a new type of distances, called transition-sensitive distances, which count, in addition to the number of mismatches, the cost of transitions between positionally adjacent match-mismatch pairs, as part of the distance. The cost of transitions that reflects the dispersion of mismatches can be integrated into conventional distance measures. We introduce transition-sensitive variants of LD and HD, referred to as TLD and THD. It is shown that while TLD and THD hold properties of the metric similarly as LD and HD, they function as more strict distance measures in similarity search applications than LD and HD, respectively.

Keywords: Transition-sensitive Distance, Transition-sensitive Levenshtein Distance, Transition-sensitive Hamming Distance, distance measure, metric, string matching, pattern matching, dynamic programming.

1 Introduction

Recently, a variety of information has been accumulated to a growing scale. Highly demanded is a simple and efficient method to retrieve relevant information of interest out of the accumulated source. In the core of information retrieval systems is data comparison or pattern matching. Numerous methods and strategies have been developed for comparison of various kinds of symbolic data, including text, voice, music, image, and video [1–7]. The relevance of the retrieved result and the efficiency of the computational process are both strongly influenced by the distance measure used for comparison.

When two patterns of equal size in arbitrary dimensions, such as multi-dimensional bitmap image data, to be compared, Hamming distance (HD) [8] has been widely used, which is defined as the minimum number of substitutions required to transform one pattern into the other. For comparison of two strings, whose lengths may be different, Levenshtein (or edit) distance (LD) [9] has been used instead, which is defined as the minimum number of insertions, deletions, and substitutions required to transform one string to the other.

Given a query string 'form', for example, two strings, 'forms' and 'forum', are mapped by LD at the same distance, 1, both for one insertion: the last character 's' of 'forms' and the fourth character 'u' of 'forum'. Mismatches, including deletions, insertions, and substitutions, break up one string into fragments, each composed of all matching (or mismatched) characters. The former 'forms' is split into two fragments of lengths 4 and 1, such as form-s, while the latter 'forum' is broken into three fragments of lengths 3, 1 and 1, such as for-u-m. Thus the LD measure gives the same distance as long as the number of mismatches is the same no matter whether the mismatches are blocked or distributed, whether they are on the edge or in the middle.

In many different applications, such as linguistic analysis, it is often presumed that strings with mismatches at the head or tail may be related objects, while those with mismatches found in the middle or distributed throughout could be independent objects. In the previous example, 'forms' (form-s) is a variant of 'form', while 'forum' (for-u-m) is an independent word. Simple methods for segregating variants from others are highly demanded for natural language processing systems as in [10].

To further differentiate those two strings (or arrays) which are mapped at the same LD (or HD), other measures need to be cascaded as additional steps. To capture the locations of mismatches, the N-gram method that is to conduct pattern matching locally in a window of length N sliding along each string has been used as in [11, 12]. To assess the degree of fragmentation of mismatches, Shannon entropy, which is defined: $H = -\Sigma p_i \cdot \log p_i$ with the occupancy p_i of a fragment of length i, has been used as in [13–15].

In this paper, we present a new type of distances, called transition-sensitive distances, that reflect not only the sum but also the distribution of mismatches between the subjects of comparison.

2 Transition-Sensitive Distances

2.1 Transition-Sensitivity

Suppose that the subjects of comparison are arrays of arbitrary number of dimensions and size. Each array is composed of symbolic elements that are either atomic or composite. Atomic elements are quantitatively comparable symbols. Composite elements are those consisting of two or more atomic elements.

The difference between two corresponding elements is referred to as the element dissimilarity. Depending on the comparison method used, whether discrete or fuzzy, the element dissimilarity may be a discrete binary integer (0 for match and 1 for mismatch) or a real number in the range [0,1] in fuzzy indicating a degree of mismatch.

The difference between two element dissimilarities at adjacent positions, which is referred to as the transition, is represented as a real number in the range

[-1, 1], where negative numbers, positive numbers, and zero represent ascents, descents and none, respectively. The positional adjacency is defined by the spatial properties of the arrays. Transition-sensitivity is the property that the distance is variable depending on the transitions in the element dissimilarities. In the following, we extend two conventional metrics, Levenshtein distance and Hamming distance, to be transition-sensitive.

2.2 Transition-Sensitive Levenshtein Distance

Transition-sensitive Levenshtein distance (TLD) is a distance between two strings, which is formulated in the dynamic programming manner similarly as Levenshtein distance (LD) is.

Definition 1 (Transition-sensitive Levenshtein Distance, TLD). *Given a string X of length m and a string Y of length n, the transition-sensitive Levenshtein distance (TLD) between the two strings is:*

$$TLD(X, Y) = D[m, n]$$

where

1. *$D[i, j]$ $(0 \le i \le m, 0 \le j \le n)$ is a string distance defined as:*

$$D[i,0] = i,\ (0 \le i \le m);\quad D[0,j] = j,\ (0 \le j \le n);$$

$$D[i,j] = min \left\{ \begin{array}{ll} D[i-1,j] + 1 + t(d[i-1,j], d[i,j], A, B) & \textit{(deletion)} \\ D[i,j-1] + 1 + t(d[i,j-1], d[i,j], A, B) & \textit{(insertion)} \\ D[i-1,j-1] + d[i,j] + t(d[i-1,j-1], d[i,j], A, B) & \textit{(substitution)} \end{array} \right\},$$

$$(0 \le i \le m, 0 \le j \le n).$$

2. *$d[i, j]$ $(0 \le i \le m, 0 \le j \le n)$ is an element dissimilarity defined as:*

$$d[0,0] = -1;\quad d[i,0] = 1,\ (1 \le i \le m);\quad d[0,j] = 1,\ (1 \le j \le n);$$
$$d[i,j] = c(X_i, Y_j),\ (1 \le i \le m, 1 \le j \le n);$$

X_i and Y_j are the i-th element of X and the j-th element of Y, respectively.

3. *$c(x, y)$ is a function that returns a real number within the range $[0, 1]$ representing the dissimilarity (or normalized distance) between two elements, x and y as follows:*

$$0 \le c(x, y) = |x - y| \le 1$$

Note that 0 indicates a complete match and 1 a complete mismatch.
The element dissimilarity may be either taken as it is or further binarized with a given threshold γ, called the dissimilarity threshold, as:

$$c(x,y) = \left\{ \begin{array}{ll} 0 \ \textit{if} \ |x-y| \le \gamma & \textit{(match)} \\ 1 \ \textit{otherwise} & \textit{(mismatch)} \end{array} \right.$$

4. *$t(d1, d2, A, B)$ is a function that returns a non-negative real number representing the cost for a transition from one element dissimilarity $d1$ to the other $d2$ as follows:*

$$t(d1, d2, A, B) = \begin{cases} A \cdot (d2 - d1) & \text{if } 0 \leq d1 < d2 \quad \text{(ascent)} \\ B \cdot (d1 - d2) & \text{if } 0 \leq d2 < d1 \quad \text{(descent)} \\ 0 & \text{otherwise} \quad \text{(no transition)} \end{cases}$$

A and B are the cost coefficient for the ascent or descent, respectively, under the following constraint:

$$0 \leq A + B \leq 1.$$

□

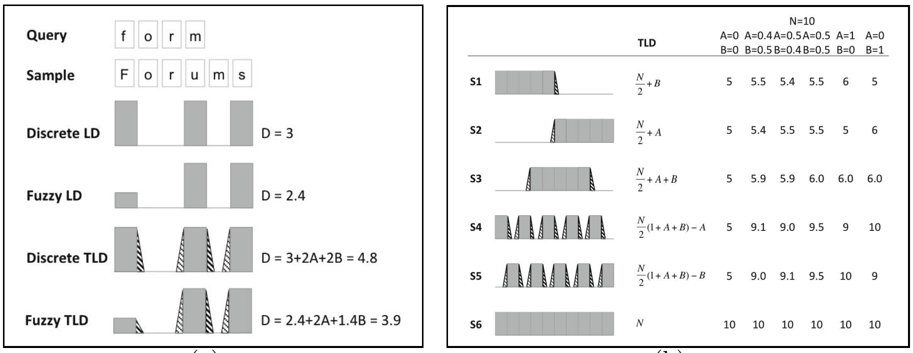
Proposition 1. *Given two strings x and y and transition cost coefficients A and B, $TLD(x, y)$ satisfies the following conditions:*

1. *$TLD(x, y) \geq 0$ (non-negativity)*
2. *$|| x | - | y || \leq TLD(x, y) \leq max(| x |, | y |)$ (lower and upper bounds)*
3. *$TLD(x, y) = 0$ if and only if $x = y$ and $| x | = | y |$ (identity)*
4. *$TLD(x, y) = TLD(x, y)$ (symmetry)*
5. *Given another string z, $TLD(x, z) \leq TLD(x, y) + TLD(y, z)$ (triangle inequality)*

where $| x |$ denotes the length of string x.

Proof.

1. $TLD(x, y)$ is defined with addition, multiplication and minimum operators on non-negative numbers, thereby resulting in a non-negative number.
2. If x and y do not match in any elements, the element dissimilarity matrix is filled up with 1s, yielding the maximum value of $TLD(x, y)$ through the shortest path of the matrix, which is equal to the larger one of the string lengths. For x and y of different lengths, the case where the shorter string fully matches either the beginning or ending part of the longer string yields the minimum value of $TLD(x, y)$, $|| x | - | y || + min(A, B)$, where $min(A, B) = 0$ if $A = B = 0$.
3. if and only if x and y are of the same length and fully match, the element dissimilarity matrix is filled up with $0s$, yielding $TLD(x, y) = 0$.
4. In computing $TLD(x, y)$ and $TLD(y, x)$, element dissimilarities and transition costs are similarly maintained, except that their coordinates are transposed. Since the computation algorithm is uniform for each axis of the coordinates, the same distance is yielded.
5. It is trivial for special cases involving identity, including $x = y = z$, $x = y \neq z$, $x \neq y = z$, and $x = z \neq y$. In other cases, there is at least one mismatch in all three paths: $x \to y$, $y \to z$ and $x \to z$. If element dissimilarities and accompanied transitions on the path $x \to y$ and those on the path $y \to z$ do not positionally overlap in y, $TLD(x, z) = TLD(x, y) + TLD(y, z)$ can hold. Otherwise, $TLD(x, z) \leq TLD(x, y) + TLD(y, z)$ holds, since positional overlaps on the sequential path $x \to y \to z$ may be reduced on the direct path $x \to z$, such as double substitutions to a single substitution or zero for the reversion. □



	TLD	N=10					
		A=0 B=0	A=0.4 B=0.5	A=0.5 B=0.4	A=0.5 B=0.5	A=1 B=0	A=0 B=1
S1	$\frac{N}{2}+B$	5	5.5	5.4	5.5	6	5
S2	$\frac{N}{2}+A$	5	5.4	5.5	5.5	5	6
S3	$\frac{N}{2}+A+B$	5	5.9	5.9	6.0	6.0	6.0
S4	$\frac{N}{2}(1+A+B)-A$	5	9.1	9.0	9.5	9	10
S5	$\frac{N}{2}(1+A+B)-B$	5	9.0	9.1	9.5	10	9
S6	N	10	10	10	10	10	10

(a) (b)

Fig. 1. Levenshtein distance (LD) and Transition-sensitive Levenshtein distance (TLD). (a) Comparison of strings, 'form' and 'Forums', with LD and TLD in discrete and fuzzy matching modes. Gray rectangles indicate element dissimilarities (or mismatches), each of which is either 0 or 1 in the discrete mode or a real number in the range [0,1] in the fuzzy mode. such as c(*Uppercase*, *Lowercase*)=0.4. Striped triangles indicate ascending or descending transitions, whose cost coefficients,A=0.4 and B=0.5, are assumed. (b) TLD for different patterns of mismatches and various cost coefficients. Note that TLD with A=0 and B=0 is equivalent to LD.

Figure 1(a) illustrates comparison of two strings, 'form' and 'Forums', using four different distance measures: Discrete LD, Fuzzy LD, Discrete TLD and Fuzzy TLD. While LD counts merely mismatches, TLD counts not only mismatches but also their transition costs. 'Discrete' or 'Fuzzy' implies whether the mismatch is represented with a binary or real number, respectively. The value of TLD varies depending on the distribution of mismatches and also on the cost coefficients for ascending and descending transitions, as shown in Figure 1(b). In the first five cases (S1-S5), there are five mismatches in common between the compared strings. However, the mismatches are distributed differently, as blocked in S1-S3 and distributed in S4-S5, so that they are given different TLDs (see the second or later columns). In the last string S6, all elements mismatched, so that LD and TLD are both 10. S4 and S5 contain only 50% mismatches, but the mismatches are evenly distributed, resulting in the highest transition cost. When the cost coefficients are set high, the TLDs of S4 and S5 get close or equal to the maximum 10 of S6 that contains 100% mismatches.

2.3 Transition-Sensitive Hamming Distance

Transition-sensitive Hamming distance (THD) is a distance between two matrices of equal size in arbitrary dimensions. Unlike LD and TLD, insertions and deletions are not allowed in HD and THD. For n-dimensional matrices, Transition-sensitive Hamming distances are formulated below.

Definition 2 (Transition-Sensitive Hamming Distance, THD). *Given two matrices X and Y of equal size $m_1 m_2 \dots m_n$ in n dimensions, n-dimensional*

Hamming distance (HDn), n-dimensional transition cost (TCn), and Transition-sensitive n-dimensional Hamming distance (THDn) between the two matrices are respectively defined as:

$$HDn(X,Y) = \sum_{i_1=1}^{m_1} \cdots \sum_{i_n=1}^{m_n} d[i_1,\ldots,i_n]$$

$$TCn(X,Y) = \sum_{i_1=1}^{m_1} \cdots \sum_{i_n=1}^{m_n} \sum_{h=1}^{n} t(d[i_1,\ldots,i_{h-1},\ldots i_n], d[i_1,\ldots,i_h,\ldots,i_n], A_h, B_h)$$

$$THDn(X,Y) = HDn(X,Y) + TCn(X,Y)$$

where

1. *$d[i_1,\ldots,i_n]$ $(1 \le h \le n,\ 0 \le i_h \le m_k)$ is an element dissimilarity defined as:*

$$d[0,\ldots,0] = -1;$$
$$d[i_1,\ldots,i_h = 0,\ldots,i_n] = 1,\ (1 \le h \le n);$$
$$d[i_1,\ldots,i_n] = c(X_{i_1,\ldots,i_n}, Y_{i_1,\ldots,i_n}),\ (1 \le h \le n,\ 1 \le i_h \le m_h);$$

   $X_{i_1,\ldots,i_n}$ and $Y_{i_1,\ldots,i_n}$ are the elements of X and Y at the corresponding position, $i_1,\ldots,i_n$, respectively.

2. *$c(x,y)$ and $t(d1,d2,A,B)$ are functions previously defined.* □

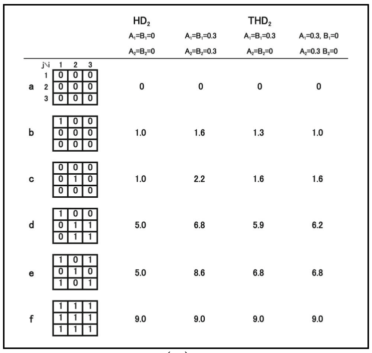
Proposition 2. *Given two matrices x and y of the same dimension N and size L_i and transition cost coefficients A_i and B_i for each dimension $(1 \le i \le N)$, $THD(x,y)$ satisfies the following conditions:*

1. *$THD(x,y) \ge 0$ (non-negativity)*
2. *$0 \le THD(x,y) \le L$ (lower and upper bounds)*
3. *$THD(x,y) = 0$ if and only if $x = y$ (identity)*
4. *$THD(x,y) = THD(x,y)$ (symmetry)*
5. *Given another matrix z of the same dimension and size as x and y, $THD(x,z) \le THD(x,y) + THD(y,z)$(triangle inequality)*

where $L = \prod_{i=1}^{n} L_i$.

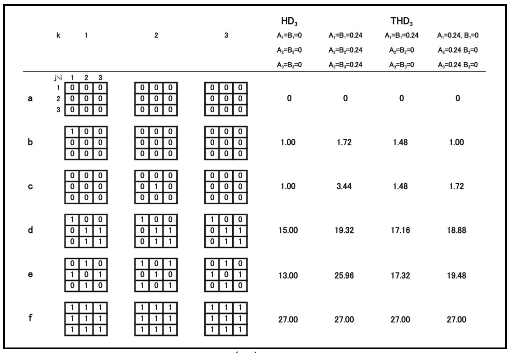
Proof. Similarly proved as in Proposition 1. □

Figures 2(a) and 2(b) illustrate THD used for comparison of two-dimensional (2D) and three-dimensional (3D) data, respectively. In individual figures, pattern a shows a complete match and pattern f a complete mismatch. In pattens b and c, there is one mismatch in common, but the mismatch in pattern c is in the center, costing more for the transitions than in pattern b where the mismatch is in the corner. In patterns d and e of Figure 2(a), there are five mismatches in common, but the mismatches in pattern e are more distributed, costing more for the transitions than the rather blocked mismatches in pattern d.



		HD_2		THD_2	
		$A_1=B_1=0$	$A_1=B_1=0.3$	$A_1=B_1=0.3$	$A_1=0.3, B_1=0$
		$A_2=B_2=0$	$A_2=B_2=0.3$	$A_2=B_2=0$	$A_2=0.3\ B_2=0$
a	$\begin{array}{c\|ccc} j\backslash i & 1 & 2 & 3 \\ \hline 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 \end{array}$	0	0	0	0
b	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	1.0	1.6	1.3	1.0
c	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	1.0	2.2	1.6	1.6
d	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$	5.0	6.8	5.9	6.2
e	$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$	5.0	8.6	6.8	6.8
f	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	9.0	9.0	9.0	9.0

(a)



	k: 1	2	3	HD_3		THD_3	
				$A_1=B_1=0$	$A_1=B_1=0.24$	$A_1=B_1=0.24$	$A_1=0.24, B_1=0$
				$A_2=B_2=0$	$A_2=B_2=0.24$	$A_2=B_2=0$	$A_2=0.24\ B_2=0$
				$A_3=B_3=0$	$A_3=B_3=0.24$	$A_3=B_3=0$	$A_3=0.24\ B_3=0$
a	$\begin{array}{c\|ccc} j\backslash i & 1 & 2 & 3 \\ \hline 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 3 & 0 & 0 & 0 \end{array}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	0	0	0	0
b	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	1.00	1.72	1.48	1.00
c	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	1.00	3.44	1.48	1.72
d	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$	15.00	19.32	17.16	18.88
e	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	13.00	25.96	17.32	19.48
f	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	27.00	27.00	27.00	27.00

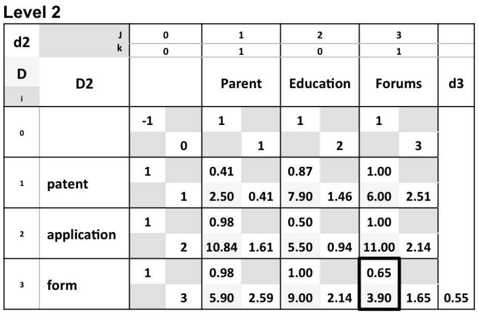
(b)

Fig. 2. Transition-sensitive Hamming Distance (THD). (a) THD for 2D data (THD2). Transitions are counted along two different axes, i and j. (b) THD for 3D data (THD3). Transitions are counted along three different axes, i, j, and k.

3 Applications

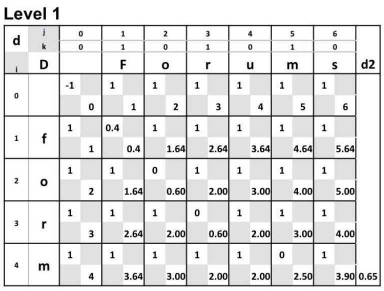
3.1 Application of TLD: Approximate Name Search

Given two strings, it is simple to compare the whole strings in the flat form, that is, in the manner of *whole-string matching*. When strings are physically large or contain semantic components, however, it is desirable to reflect the structures of the strings in the comparison. In the case of natural language texts, for example, statements can be split into coarse-grain elements (e.g., words) and further into fine-grain elements (e.g., characters). TLD is applicable to the comparison at such different grain levels.



Level 2

d2	J	0		1		2		3		
	k	0		1		0		1		
D i	D2			Parent		Education		Forums		d3
0		-1		1		1		1		
			0		1		2		3	
1	patent	1		0.41		0.87		1.00		
			1	2.50	0.41	7.90	1.46	6.00	2.51	
2	application	1		0.98		0.50		1.00		
			2	10.84	1.61	5.50	0.94	11.00	2.14	
3	form	1		0.98		1.00		0.65		
			3	5.90	2.59	9.00	2.14	3.90	1.65	0.55



Level 1

d	j	0		1		2		3		4		5		6		
	k	0		1		0		1		0		1		0		
i	D			F		o		r		u		m		s		d2
0		-1		1		1		1		1		1		1		
			0		1		2		3		4		5		6	
1	f	1		0.4		1		1		1		1		1		
			1		0.4		1.64		2.64		3.64		4.64		5.64	
2	o	1		1		0		1		1		1		1		
			2		1.64		0.60		2.00		3.00		4.00		5.00	
3	r	1		1		1		0		1		1		1		
			3		2.64		2.00		0.60		2.00		3.00		4.00	
4	m	1		1		1		1		1		0		1		
			4		3.64		3.00		2.00		2.00		2.50		3.90	0.65

Fig. 3. Hierarchical application of TLD. Strings, 'patent application form' and 'Patent Education Forums', are split into words (Level 2), where a single space is used as the separator. The word distance D (at the left bottom) obtained from word comparison (Level 1) is normalized to the dissimilarity $d2$ (at the left top) to be used for phrase comparison (Level 2), i.e., for words u and v, $d2(u, v) = D(u, v)/max(\mid u \mid, \mid v \mid)$. The phrase distance $D2$ (at the right bottom) is similarly normalized to the dissimilarity $d3$. Note that the transition cost coefficients, A=A2=0.4 and B=B2=0.5, and the element dissimilarity, c(*Uppercase, Lowercase*)=0.4, are assumed.

Figure 3 introduces a hierarchical method of string comparison, referred to as the *word-wise matching*. After strings, 'patent application form' and 'Parent Education Forums', are individually split into words, TLD is similarly applied to both word comparison (Level 1) and phrase comparison (Level 2). The word distance D obtained from word comparison is normalized to the dissimilarity $d2$ to be used for phrase comparison. After conducting phrase comparison similarly as word comparison, the obtained phrase distance $D2$ is normalized to the dissimilarity $d3$.

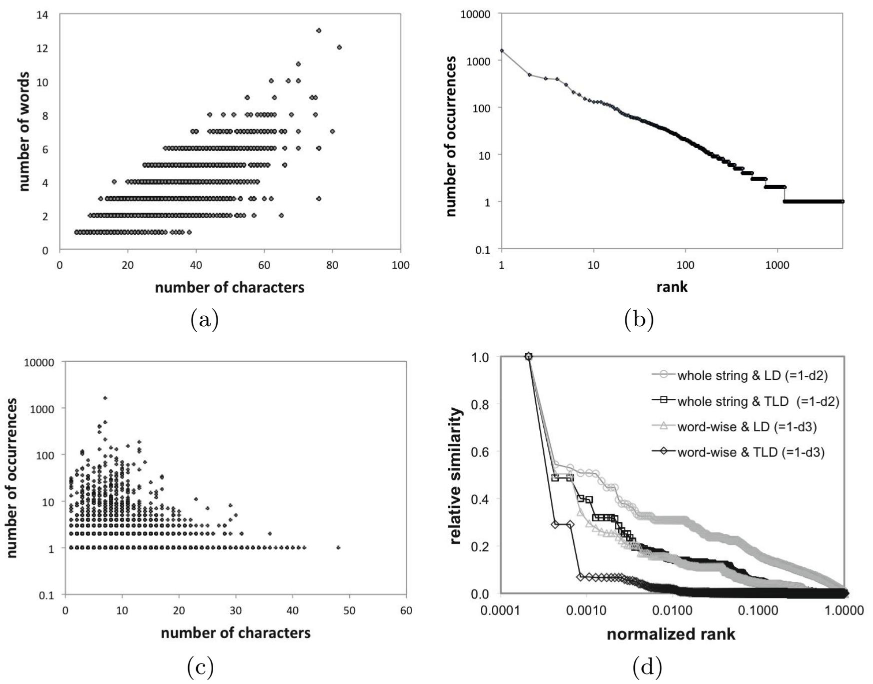


Fig. 4. Approximate String Search. Total 4688 records of biological terms in a database were compared against the query string, 'ribosomal RNA processing', using two different matching methods (*whole-string* and *word-wise*) in combination with two different distance measures (LD and TLD). (a) Composition of individual records in the database. (b) Frequency of individual words constituting the database. (c) Frequency versus composition of individual words. (d) Relative similarity of individual records to the query, evaluated through four different measures, where the transition cost coefficients, A=0.4 and B=0.5, and the element dissimilarity, c(*Uppercae, Lowercase*)=0.4, are assumed.

Using two different matching methods (*whole string* and *word-wise*) in combination with two different distance measures (LD and TLD), we conducted approximate name search on a database containing 4688 different biological terms, including gene and protein names. The content of the database is statistically characterized as shown in Figures 4(a) - 4(c). Individual string records were composed of 5-82 (mean 31) characters and of 1-13 (mean 3) words as shown in Figure 4(a). Total 5115 different words were contained in the database. As shown in Figure 4(b), the frequency of words made a power-law distribution, obeying Zipf's law [16] that is empirically known to hold for natural language texts rather than for artificial language texts. Frequently appearing words were general terms of 3-13 characters, such as 'protein', 'subunit and 'domain' ranked in the top, as shown in Figure 4(c). A string, 'ribosomal RNA processing', composed of 24 characters containing three words, was used as the query. While the query string had a complete match with only one record, individual words frequently appeared in the database, such that 'ribosomal' occurred in 129 records, 'RNA' in 117 records and 'processing' in 33 records.

After comparing individual string records against the query string, the dissimilarity obtained for each string record i was converted to the similarity and normalized against the maximum of similarity to the relative similarity: $S_i = 1 - d*_i$; $s_i = S_i / max(S)$. The relative similarity was ranked and normalized against the total number of records (N) to the relative rank: $r_i = rank(S_i)/N$. Functions of the relative similarity to the normalized rank, evaluated through the four different measures, are plotted in Figure 4(d). The inner the function curve lies, less records will chosen above the given threshold, so that the more strict the evaluation is meant to be. For each matching method, the curve of TLD was found inner than that of LD. For each distance measure, the curve of the word-wise matching was found inner than that of the whole-string matching. Given 0.4 relative similarity as the threshold, for example, only one record was fished by the word-wise matching with TLD, while 4, 3 and 10 records were found by the whole string matching with TLD, the word-wise matching with LD and the whole string matching with LD, respectively. Thus, the combination of the word-wise matching and TLD is suggested to be the most strict evaluation measure that would retrieve the least number of records as those above the given threshold of similarity.

3.2 Application of THD: Image Clustering

THD is applicable to comparison of multi-dimensional data, such as images and volumes, similarly as HD is. To see how differently THD and HD may behave, we conducted pair-wise comparison on 20 different images shown in Figure 5(a), using each of the two distance measures. Individual images are 150x150 black&white pixels in resolution and statistically characterized with their darkness (the ratio of black pixels to the entire image) and normalized standard deviation (the standard deviation of the distances of black pixels from their center, normalized against the maximum of standard deviation), as shown in Figure 5(b).

For the 20 different images, total 190 pair-wise distances were computed with HD or THD as the distance measure, and normalized to dissimilarities to produce a distance matrix. The 190 pair-wise dissimilarities (d) were converted to similarities ($s = 1 - d$) and plotted against their ranks in Figure 5(c). The curve for THD lies inner and more sharply declines than the one for HD does,

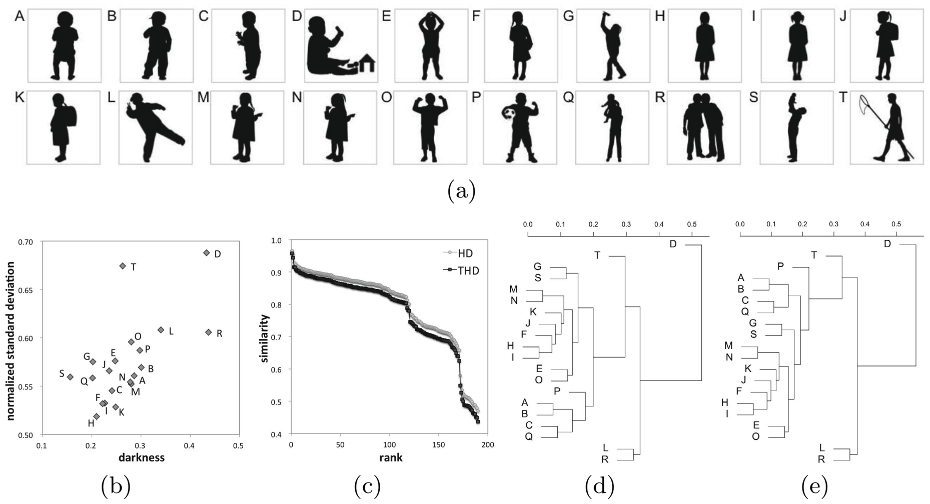


Fig. 5. Image Clustering. (a) 20 images (A-T) used in the study. (b) Darkness and normalized standard deviation of individual images. (c) Similarities ($s = 1 - d$) of 190 different pairs of images, where HD (gray circles) and THD (black squares) were used for computation of dissimilarities (d). (d-e) Dendrograms obtained through hierarchical complete-linkage clustering conducted on the distance matrix of the images, using (d) HD or (e) THD as the distance measure. Note that the transition cost coefficients, A=0.5 and B=0.5 were used in the computation of THD.

suggesting that less candidates would be retrieved with THD than with HD, given a certain similarity as the threshold.

Merely the values of similarities decreased with THD? To see if there is any change in the relationship of proximity among the images, we conducted hierarchical complete-linkage clustering [17] on individual distance matrices produced with HD and THD. Figures 5(d) and 5(e) show the resulting dendrograms for HD and THD, respectively. Images D, L, R and T, which are clearly separated from the rest in Figure 5(b), are segregated from the rest similarly in both dendrograms. The difference is in the rest. While image P is grouped together with images A, B, C and Q in one clade and the rest forms another clade in the dendrogram with HD (Figure 5(d)), image P is segregated from images A, B, C and Q and the whole rest are placed in one clade in the dendrogram with THD (Figure 5(e)). Other than image P, the relationship of proximity is maintained in both HD and THD. Thus, the cost of transitions introduced in THD seemingly made the overall evaluation of similarity more strict and contributed to the differentiation of a rather dense clade.

4 Concluding Remarks

In this paper, we presented a new type of distances, called transition-sensitive distances, which count, in addition to the number of mismatches between the compared data, the cost of transitions between positionally adjacent match-mismatch pairs, as part of the distance. By integrating the cost of transitions, conventional distance measures can be extended to be transition-sensitive.

We introduced transition-sensitive variants of LD and HD, referred to as TLD and THD. Compared with LD (or HD), each unit operation of TLD (or THD) needs one additional step of computation for the cost of transitions, but the computational order remains the same. TLD for strings of length m and n is of the order of $O(mn)$. 1D (or 2D) THD for strings of length n (or objects of size $m \times n$) is of the order of $O(n)$ (or $O(mn)$). Also, one additional matrix to store element dissimilarities is required for TLD and 2D THD, but it is possible to recycle a matrix of two rows in the actual implementation since only adjacent rows are used in the computation. We showed that while TLD and THD hold properties of the metric as well as LD and HD, they function as more strict distance measures in similarity search applications than LD and HD, respectively.

Properties of the cost of transitions are similar to those of entropy that has been widely used as a measure of randomness (or the amount of information) in various applications, including encoding [18], music analysis [19], linguistic analysis [20], and bioinformatics [21]. The more disperse the mismatches are, the higher the cost of transitions will be. If the transition cost coefficients are set to occupy one mismatch, the cost of transitions is highest when the matches and a mismatches are alternated. When the ratio of mismatches is either smaller or larger than 0.5, the chance of getting transitions is less. Unlike entropy that is defined uniformly throughout the data space, the cost of transitions is defined with two separate coefficients, one for ascends and the other for descends. The separation of cost coefficients makes it possible to express the locational allowance on mismatches, i.e., whether mismatches are more acceptable in the leading or tailing part. Behaviors of transition-sensitive distances depending on various cost coefficients remain to be elucidated in the future work.

In summary, the essence of transition-sensitive distances is that the cost of transitions is embedded as part of a distance, rather than regarded as an orthogonal measure that should be independently applied. At the cost of precision as a measure due to the dimensionality reduction, gained is computational simplicity that is required for a large scale of data mining, classification and machine learning. Transition-sensitive distances enable one to segregate data based on the number and dispersion of mismatches in a single pass of computation, so that they are useful for screening large datasets or information streams to retrieve and classify those objects that are highly similar to some part of the target. As more information accumulates and flows, more needs for transition-sensitive distances will rise in various fields of similarity search and applications.

References

1. Navarro, G.: A guided tour to approximate string matching. ACM Computing Surveys 33(1), 31–88 (2001)
2. Cohen, W.W., Ravikumar, P., Fienberg, S.E.: A comparison of string distance metrics for name-matching tasks. In: Proceedings of the ACM Workshop on Data Clearning, Record Linkage and Object Identification (2003)

3. Liu, C.-C., Hsu, J.-L., Chen, A.L.P.: An approximate string matching algorithm for content-based music data retrieval. In: Proceedings of the IEEE International Conference on Multimedia Computing and Systems, vol. 2, p. 9451 (1999)
4. Clifford, R., Iliopoulos, C.: Approximate string matching for music analysis. Soft Computing - A Fusion of Foundatios, Methodologies and Applications 8(9), 597–603 (2004)
5. Yeh, M.-C., Cheng, K.-T.: A string matching approach for visual retrieval and classification. In: Proceeding of the 1st ACM Conference on Multimedia Information Retrieval, pp. 52–58 (2008)
6. Adjeroh, D.A., Lee, M.C., King, I.: A distance measure for video sequences. Computer Vison and Image Understanding 75(1/2), 25–45 (1999)
7. Bezerra, F.N., Leite, N.J.: Using string matching to detect video transitions. Pattern Analysis & Applications 10(10), 45–54 (2007)
8. Hamming, R.W.: Error detecting and error correcting codes. Bell System Technical Journal 29(2), 147–160 (1950)
9. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics=Doklady, Cybernetics and Control Theory 10(8), 707–710 (1966)
10. Zelenko, D.: System and method for variant string matching. World Intellectual Property, WO/2009/094649, PCT/US2009/032034 (2009)
11. Gravano, L., Ipeirotis, P.G., Jagadish, H.V., Koudas, N., Murthukrishnan, S., Pietarinen, L., Srivastava, D.: Using q-grams in a DBMS for approximate string processing. IEEE Data Engineering Bulletin 24, 28–34 (2001)
12. Wang, C., Li, J., Shi, S.: N-gram inverted index structures on music data for theme mining and content-basd information retrieval. Pattern Recognition Letters 27(5), 492–503 (2006)
13. Shannon, C.E.: A mathematical theory of communication. Bell System Technical Journal 27, 379–423, 623–656 (1948)
14. Shannon, C.E.: Prediction and entropy of printed english. Bell System Technical Journal 30, 50–64 (1951)
15. Lin, J.: Divergence measures based on the Shannon entropy. IEEE Transactions on Information Theory 37(1), 145–151 (1991)
16. Zipf, G.K.: Human behavior and the principle of least effort. Addison-Wesley, Cambridge (1949)
17. Defays, D.: The efficient algorithm for a complete link method. The Computer 20(4), 364–366 (1977)
18. Yang, S.: Entropy distance. Computing Research Repository, 1303.0070 (2013)
19. Camarena-Ibarrola, A., Chávez, E.: On musical performances identification, entropy and string matching. In: Gelbukh, A., Reyes-Garcia, C.A. (eds.) MICAI 2006. LNCS (LNAI), vol. 4293, pp. 952–962. Springer, Heidelberg (2006)
20. Juola, P.: Cross-entropy and linguistic typology. In: Powers, D.M.W. (ed.) NeMLaP3CoNLL98: New Methods in Language Processing and Computational Natural Language Learning, pp. 141–149. ACL
21. Benson, G.: A new distance measure for comparing sequence profiles based on path lengths along an entropy surface. Bioinformatics 18(suppl. 2), S44–S53 (2002)