COMMUNICATIONS IN INFORMATION AND SYSTEMS
Vol. 5, No. 4, pp. 367-384, 2005

001

COMMON PITFALLS USING THE NORMALIZED COMPRESSION DISTANCE: WHAT TO WATCH OUT FOR IN A COMPRESSOR

MANUEL CEBRIÁN*, MANUEL ALFONSECA*, AND ALFONSO ORTEGA*

Abstract. Using the mathematical background for algorithmic complexity developed by Kolmogorov in the sixties, Cilibrasi and Vitanyi have designed a similarity distance named normalized compression distance applicable to the clustering of objects of any kind, such as music, texts or gene sequences. The normalized compression distance is a quasi-universal normalized admissible distance under certain conditions. This paper shows that the compressors used to compute the normalized compression distance are not idempotent in some cases, being strongly skewed with the size of the objects and window size, and therefore causing a deviation in the identity property of the distance if we don't take care that the objects to be compressed fit the windows. The relationship underlying the precision of the distance and the size of the objects has been analyzed for several well-known compressors, and specially in depth for three cases, bzip2, gzip and PPMZ which are examples of the three main types of compressors: block-sorting, Lempel-Ziv, and statistic.

1. Introduction. A natural measure of similarity assumes that two objects x and y are similar if the basic blocks of x are in y and vice versa. If this happens we can describe object x by making reference to the blocks belonging to y, thus the description of x will be very simple using the description of y.

This is partially what a compressor does to code the catenated xy sequence: a search for information shared by both sequences in order to reduce the redundancy of the whole sequence. If the result is small, it means that a lot of information contained in x can be used to code y, following the similarity conditions described in the previous paragraph. This was formalized by Rudi Cilibrasi and Paul Vitányi [2], giving rise to the concept of *normalized compression distance* (NCD), which is based on the use of compressors to provide a measure of the similarity between the objects. This distance may then be used to cluster those objects.

This idea is very powerful, because it can be applied in the same way to all kind of objects, such as music, texts or gene sequences. There is no need to use specific features of the objects to cluster. The only thing needed to compute the distance from one object x to another object y, is to measure the ability of x to turn the description of y simple and vice versa.

Cilibrasi and Vitányi have perfected this idea in two ways, by stating the conditions that a compressor must hold to be useful in the computation of the NCD, and by giving formal expression to the quality of the distance in comparison with an ideal distance proposed by Vitányi and others in [3]. In this paper we show that the

*Departamento de Ingeniería Informática, Escuela Politécnica Superior, Universidad Autónoma de Madrid, E-mail: {manuel.cebrian, manuel.alfonseca, alfonso.ortega}@uam.es

compressor must be invariant with respect to the size of the objects. This condition doesn't hold for some well-known compressors such as bzip2, gzip, pkzip and many others if the object size exceeds the window size. However, as shown by our results, in the range of usefulness of these compressors, the NCD is very good for its purposes.

We determine the precision up to which $\mathrm{NCD}(x,y) = 0 \iff x = y$ holds for different compressors. For compressors using a certain window size, or block size, we obtain $\mathrm{NCD}(x,x)$ close to 1 once we significantly exceed the window size, as the compressors no longer compress. Trivially, in computing the $\mathrm{NCD}(x,y)$ the concatenation xy should comfortably fit the window size or block size. Note that the behavior on (x,x) is possibly different from that on (x,y), with respect to window size. Namely, a window of size $|x|$ sliding over xx has mostly all of x in the window (suffix of first instance, prefix of the next instance). The way in which the identity (of the metric) and the idempotency (of the compressor) are related is the following:

$$x = y \Rightarrow C(xy) - C(x) = O(\log|x|) \Rightarrow \mathrm{NCD}(x,y) = O\left(\frac{\log|x|}{C(x)}\right) \xrightarrow[|x|\to\infty]{} 0$$

These deficiencies observed when measuring identical objects (the easiest scenario) are obviously generalized to any pair of objects. In this way, speaking about identity-idempotency problems is the same as speaking about deficiencies in the whole distance.

In sections 2 and 3 we summarize the formalism around the distance: the conditions that the compressors must hold and the properties of the distance. Section 4 describes the materials we have used to perform our experiments. Section 5 presents our results for the bzip2 and gzip compressors, and the anomalous behavior of the distance is analyzed in detail. Finally, in our conclusions, we provide empirical advice for the correct use of the NCD.

2. Definitions. The following definitions describe the conditions under which the NCD is a quasi-universal normalized admissible distance.

DEFINITION **1.** *A compressor C is* normal *if it satisfies, up to an additive $O(\log n)$ term, with n the maximal binary length of an element involved in the (in)equality concerned, the following axioms:*

1. Idempotency: $C(xx) = C(x)$, *and* $C(\lambda) = 0$, *where λ is the empty string.*
2. Monotonicity: $C(xy) \geq C(x)$.
3. Symmetry: $C(xy) = C(yx)$.
4. Distributivity: $C(xy) + C(z) \leq C(xz) + C(yz)$.

DEFINITION **2.** *A distance $d(x,y)$ is a* normalized admissible distance *or* similarity distance *if it takes values in $[0,1]$ and satisfies the following conditions for all objects x, y, z:*

1. Identity: $d(x,y) = 0$ *if* $x = y$.
2. Symmetry: $d(x,y) = d(y,x)$.

3. Triangular inequality: $d(x,y) \leq d(x,z) + d(z,y)$.

4. Density constraint *as in [2].*

DEFINITION **3.** *The* Conditional Kolmogorov Complexity $K(x|y)$ *is the length of the shortest binary program that, on a fixed universal Turing machine, outputs* x *and halts, when it is fed with input* y*. The* Kolmogorov Complexity $K(x)$ *is the Conditional Kolmogorov Complexity for* $y = \lambda$*.* $K(x|y)$ *and* $K(x)$ *are both incomputable. Reference [4] provides a more detailed treatment of algorithmic information theory.*

DEFINITION **4.** *A normalized admissible distance* f *is* quasi-universal *if for every computable normalized admissible distance* d *and for all sequences* x, y *it satisfies the following condition:*

$$f(x,y) = O(d(x,y)), \tag{1}$$

which means that two objects (of any kind) are similar (i.e. they have a small distance) with respect to a specific feature (pitch, sequence alignment, etc) when they are also similar with respect to a quasi-universal distance.

3. Normalized Compression Distance. A universal distance is the final goal for universal clustering, because in principle it will be as good as any other distance specialized in measuring some specific feature.

Reference [3] proposes an incomputable distance that fulfills that goal, the normalized information distance (NID):

$$\mathrm{NID}(x,y) = \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}} \tag{2}$$

Inspired by this incomputable distance, the following normalized compression distance was designed, which would make the role of a quasi-universal distance:

$$\mathrm{NCD}(x,y) = \frac{\max\{C(xy) - C(x), C(yx) - C(y)\}}{\max\{C(x), C(y)\}} \tag{3}$$

$C(xy)$ is the compressed size of the catenation of x and y. NCD generates a non-negative number $0 \leq \mathrm{NCD}(x,y) \leq 1$. Distances near 0 indicate similarity between objects, while those near 1 they reveal dissimilarity. If $x = y$, NCD becomes

$$\mathrm{NCD}(x,x) = \frac{C(xx) - C(x)}{C(x)} \tag{4}$$

As mentioned in the introduction, there exists an obvious linkage between the idempotency and the identity properties. Assume the compressor is normal. By the idempotency property (Definition 1) $C(xx) - C(x) = O(\log |x|)$ and thus replacing in the Equation 6 results:

$$\mathrm{NCD}(x,x) = O\left(\frac{\log|x|}{C(x)}\right) \xrightarrow[|x|\to\infty]{} 0 \tag{5}$$

which means that if the sequence is large enough and the idempotency property holds up to a logarithmic term then the identity property is preserved. The next lines summarize the most important results proved on this distance (the proofs are available in [2]).

THEOREM **1.** *If the compressor is normal, the NCD is a normalized admissible distance satisfying the metric inequalities.*

THEOREM **2.** *Let d be a normalized admissible distance and C be a normal compressor. Then, $NCD(x,y) \leq \alpha d(x,y) + \epsilon$ where α and ϵ are well-defined constants (the details can be found in [2]).*

The last theorem states that, if the compressor is chosen carefully, then the NCD is a quasi-universal normalized distance, a golden standard for clustering.

4. Materials. The remainder of the paper analyzes the behavior of two real implementations of the distance. The CompLearn toolkit[1] is a package implemented by Rudy Cilibrasi for clustering purposes. The latest version of this package (0.6.2)[2] was used in our experiments. The bzip2 and the gzip compressors can be selected in the toolkit. Our results cover both.

On the other hand, our experimental set is the well known Calgary Corpus, a benchmark for compression algorithms since 1989. Nine different types of text are represented, and to confirm that the performance of schemes is consistent for any given type, many of the types have more than one representative.

Normal English, both fiction and non-fiction, is represented by two books and six papers (labeled book1, book2, paper1, paper2, paper3, paper4, paper5, paper6). More unusual styles of English writing are found in a bibliography (bib) and a batch of unedited news articles (news). Three computer programs represent artificial languages (progc, progl, progp). A transcript of a terminal session (trans) is included to indicate the increase in speed that could be achieved by applying compression to a slow line in a terminal. All of the files mentioned so far use ASCII encoding. Some non-ASCII files are also included: two files of executable code (obj1, obj2), some geophysical data (geo), and a bit-map black and white picture (pic). File geo is particularly difficult to compress, because it contains a wide range of data values, while file pic is highly compressible, because of large amounts of white space in the picture, represented by long runs of zeros. More reasons for choosing this benchmark are explained in reference [5].

[1] All the experiments published in [2] were performed using this toolkit.

[2] Available in the Internet at http://www.complearn.org

TABLE 1
Name, description and size in Kbytes of the files in the Calgary Corpus.

bib	Bibliographic files (refer format)	109
book1	Hardy: Far from the madding crowd	751
book2	Witten: Principles of computer speech	597
geo	Geophysical data	100
news	News batch file	369
obj1	Compiled code for Vax: compilation of progp	21
obj2	Compiled code for Apple Macintosh: Knowledge support system	242
paper1	Witten, Neal and Cleary: Arithmetic coding for data compression	52
paper2	Witten: Computer (in)security	81
paper3	Witten: In search of "autonomy"	46
paper4	Cleary: Programming by example revisited	13
paper5	Cleary: A logical implementation of arithmetic	12
paper6	Cleary: Compact hash tables using bidirectional linear probing	38
pic	Picture number 5 from the CCITT Facsimile test files (text + drawings)	502
progc	C source code: compress version 4.0	39
progl	Lisp source code: system software	70
progp	Pascal source code: prediction by partial matching evaluation program	49
trans	Transcript of a session on a terminal	92

5. Results. In our experiments, all the objects are considered strings of bytes. If x is an object, then x_n is the object composed by the first n bytes of x. Figures 1 and 9 show the distance $\mathrm{NCD}(x_n, x_n)$ as a function of n in four (bib, book1, book2, pic) of the eighteen files of the Calgary Corpus[3]. The files book1 and book2 are selected to be shown because they are the larger ones, while the file bib is selected because of its average size. The reason for showing the file pic is because it is a large but highly compressible file. To analyze the idempotency property, all the objects are compared with themselves.

5.1. bzip2. The plots in Figure 1, together with many more similar experiments we have performed, show that $\mathrm{NCD}(x, x)$ is between 0.2 and 0.3 in the region where bzip2 can be used properly, while it gives values between 0.25 and 0.9 outside that region.

Two plots in Figure 1 (book1 and book2) show two visually different modes of dependency as a function of n. We call *weak dependency* the region starting in 1 Kbytes and ending in 450 Kbytes (the bib and pic files only show this dependency because of the small size of the first one and the high compressibility of the second one). From that point onwards we call it *strong dependency*.

The weak dependency displays a fluctuating slowly-increasing dependence with

[3]The reason for choosing only four of the eighteen files is purely aesthetic. The remaining (not displayed) graphs are available by request to the authors.

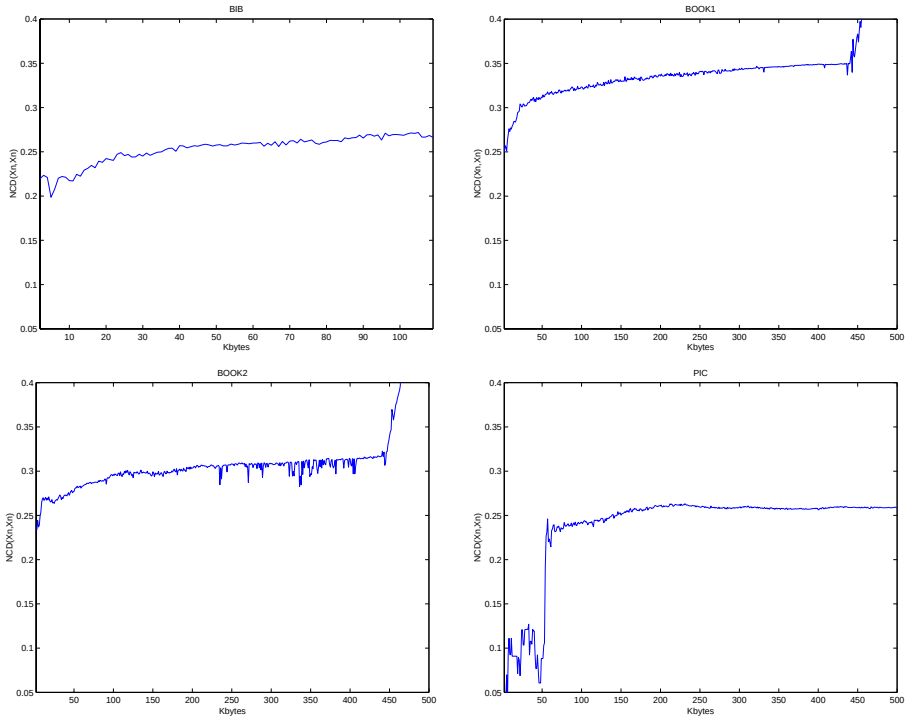

FIG. 1. *Normalized compression distances computed for the first n bytes of four files (bib, book1, book2 and pic, from left to right and top to bottom) of the Calgary Corpus files using the bzip2 compressor with the --best option.*

n. On the other hand, the strong dependence is logarithmic and almost without fluctuations. Both dependencies indicate that the distance is skewed by the size of the objects and therefore displays idempotency-identity deviations (see Definitions 1 and 2).

The bzip2 compression algorithm uses three main ideas. In the first stage of the compression, the data suffers a *Burrows-Wheeler transform*; in the second, the *move to front* coding is applied to the output of the transformation; finally a *statistical compressor* (usually Huffman) is used for redundancy extraction. The default block size of the bzip2 compressor is 900 Kbytes which means that, if the sizes of the objects add to more than 900 Kbytes, the catenated object is divided into parts smaller than 900 Kbytes before being compressed. A more detailed explanation of the algorithms in bzip2 can be found in [6].

Let's start with the weak dependency, which can be observed in the [1 Kbyte, 450 Kbytes] interval, exactly the half size of the block. In this zone, the size of the catenated objects is smaller than 900 Kbytes, thus they don't need to be split.

A simplified example will show how the weak dependency works. Let us assume that the block size is 16 bytes and the object to compress is the string "drdobbs". We

d	r	d	o	b	b	s	d	r	d	o	b	b	s
r	d	o	b	b	s	d	r	d	o	b	b	s	d
d	o	b	b	s	d	r	d	o	b	b	s	d	r
o	b	b	s	d	r	d	o	b	b	s	d	r	d
b	b	s	d	r	d	o	b	b	s	d	r	d	o
b	s	d	r	d	o	b	b	s	d	r	d	o	b
s	d	r	d	o	b	b	s	d	r	d	o	b	b
d	r	d	o	b	b	s	d	r	d	o	b	b	s
r	d	o	b	b	s	d	r	d	o	b	b	s	d
d	o	b	b	s	d	r	d	o	b	b	s	d	r
o	b	b	s	d	r	d	o	b	b	s	d	r	d
b	b	s	d	r	d	o	b	b	s	d	r	d	o
b	s	d	r	d	o	b	b	s	d	r	d	o	b
s	d	r	d	o	b	b	s	d	r	d	o	b	b

FIG. 2. *Rotation matrix for "drdobbsdrdobbs".*

need to compute the distance:

$$\text{NCD}(\text{drdobbs}, \text{drdobbs}) = \frac{C(\text{drdobbsdrdobbs}) - C(\text{drdobbs})}{C(\text{drdobbs})} \tag{6}$$

The size of the catenated string is 14 bytes, so it fits in a single block. Let's analyze the algorithm step by step:

Burrows-Wheeler transform: A rotation matrix is created from the string "drdobbsdrdobbs" (Figure 2). It can be observed that the lower half of the matrix is a repetition of the upper half. Then the matrix is lexicographically sorted and the output for the transformation is the last column of the matrix "oobbrrssddddbb" (Figure 3).

move to front coding: the coding is applied and the output is "20103040400030" (see [6]).

Huffman coding: The frequencies of the characters are measured as 0:8, 1:1, 2:1, 3:2, 4:2 and the compressed string is built using 26 bits (see [7]).

Using the same scheme, the string "drdobbs" is compressed using 17 bits, so the distance is $\text{NCD} = \frac{26-17}{17} = 0.529$.

Now another symbol "w" is added to the string, so that the new string whose distance with itself we want to measure is "drdobbsw". When the rotation matrix of "drdobbswdrdobbsw" is built and ordered, the first row whose last column has the "w" value will be followed by another row that ends in "w" (in fact both rows will be identical).

b	b	s	d	r	d	o	b	b	s	d	r	d	o
b	b	s	d	r	d	o	b	b	s	d	r	d	o
b	s	d	r	d	o	b	b	s	d	r	d	o	b
b	s	d	r	d	o	b	b	s	d	r	d	o	b
d	o	b	b	s	d	r	d	o	b	b	s	d	r
d	o	b	b	s	d	r	d	o	b	b	s	d	r
d	r	d	o	b	b	s	d	r	d	o	b	b	s
d	r	d	o	b	b	s	d	r	d	o	b	b	s
o	b	b	s	d	r	d	o	b	b	s	d	r	d
o	b	b	s	d	r	d	o	b	b	s	d	r	d
r	d	o	b	b	s	d	r	d	o	b	b	s	d
r	d	o	b	b	s	d	r	d	o	b	b	s	d
s	d	r	d	o	b	b	s	d	r	d	o	b	b
s	d	r	d	o	b	b	s	d	r	d	o	b	b

FIG. 3. *Lexicographically ordered rotation matrix for "drdobbsdrdobbs".*

Just by looking at the string (even without constructing the rotation matrix) we can see that, when the string is coded using move to front, the second "w" of "drdobbswdrdobbs*w*" will get the value 0. This means that the cost of adding "w" to the string will be the cost of coding the first "w" plus the cost of coding the second (only one bit), due to the symmetry of the rotation matrix.

In this way, when we add a symbol π to a string x giving y, the expected difference $C(yy) - C(y)$ will be larger that the expected difference $C(xx) - C(x)$ by just one bit. The codings should not differ too much, because the information of the symbol is the same in both strings, $log(\frac{1}{p(\pi)})$ in Shannon terms. This explains that the weak dependency increases very slowly with n and fluctuates.

In the example, C(drdobbswdrdobbsw) = 33, i.e. after adding two "w" to the original string (16 bits) the size of the compressed version only increases by 7 bits. The new distance is NCD = $\frac{33-22}{22} = 0.5$, almost identical to that in the original string. The compressor has noticed that the second half of the string is identical to the first, not in a direct way, but by detecting the redundancy of the second half of the string, and therefore coding that half with very few bits.

Let's now explain the strong dependency. The objects in this zone have a size greater than 450 Kbytes, therefore the catenated object has a size greater than the block size of bzip2 (900 Kbytes). In our explanation, we will assume that the block size is 8 bytes, and will use a string that, catenated to itself, is bigger than that size. We use the string "drdobbs" again, so when compressing "drdobbsdrdobbs" it must be split in the two strings "drdobbsd" and "rdobbs". Let's apply the compression

d	r	d	o	b	b	s	d
r	d	o	b	b	s	d	d
d	o	b	b	s	d	d	r
o	b	b	s	d	d	r	d
b	b	s	d	d	r	d	o
b	s	d	d	r	d	o	b
s	d	d	r	d	o	b	b
d	d	r	d	o	b	b	s

FIG. 4. *Rotation matrix for "dr-dobbsdrd".*

r	d	o	b	b	s
d	o	b	b	s	r
o	b	b	s	r	d
b	b	s	r	d	o
b	s	r	d	o	b
s	r	d	o	b	b

FIG. 5. *Rotation matrix for "rdobbs".*

algorithm to both:

Burrows-Wheeler transform: The rotation matrix for the two strings (Figures 4 and 5) is built and ordered (Figures 6 and 7). There is a big difference between having the whole string in one block or in two: the upper-lower half symmetry is lost due to the splitting, and much redundancy achieved in the weak dependence zone is not achieved here. The outputs are "obsrdddb" and "obrdsb", whose equal characters are much less grouped[4] than in the previous example "oobbrrssddddbb".

move to front coding: The coding of both strings is "21444003" and "213343".

Huffman coding: The character frequencies are 0:2, 1:1, 2:1, 3:1, 4:3 for the first string and 1:1, 2:1, 3:3, 4:1 for the second one. The resulting output built using Huffman coding has a size of 30 bits (4 more than when using a single block). The main question is not the final size of the string, but the fact that the second half of the string has been coded using 16 bits, while it was coded using only 8 bits in the single block example (exactly double). Splitting the string has caused a worse character-grouping when the Burrows-Wheeler transform is performed. The long distance redundancies of the string have been lost, and this introduces a skew in the distance: the same objects are now farther apart: NCD $= \frac{30-17}{17} = 0.765$.

If the string is divided in two blocks, the expected compression cost of adding a symbol π to x will be $2\log(\frac{1}{p(\pi)})$, therefore an expected increase of $log(\frac{1}{p(\pi)})$ in $C(xx) - C(x)$. This explains the logarithmic growth of the strong dependency zone.

In order to compare both dependencies, it should be remembered that the expected compression cost of adding one symbol is $log(\frac{1}{p(\pi)}) + 1$, therefore the expected value of $C(xx) - C(x)$ is 1 bit when the concatenated string fits into a single block.

We have repeated our experiments with bzip2 in a different situation, by selecting

[4]Grouping identical characters is the main purpose of the Burrows-Wheeler transform.

b	b	s	d	d	r	d	o
b	s	d	d	r	d	o	b
d	d	r	d	o	b	b	s
d	o	b	b	s	d	d	r
d	r	d	o	b	b	s	d
o	b	b	s	d	d	r	d
r	d	o	b	b	s	d	d
s	d	d	r	d	o	b	b

FIG. 6. *Lexicographically ordered rotation matrix for "drdobbsdrd".*

b	b	s	r	d	o
b	s	r	d	o	b
d	o	b	b	s	r
o	b	b	s	r	d
r	d	o	b	b	s
s	r	d	o	b	b

FIG. 7. *Lexicographically ordered rotation matrix for "rdobbs".*

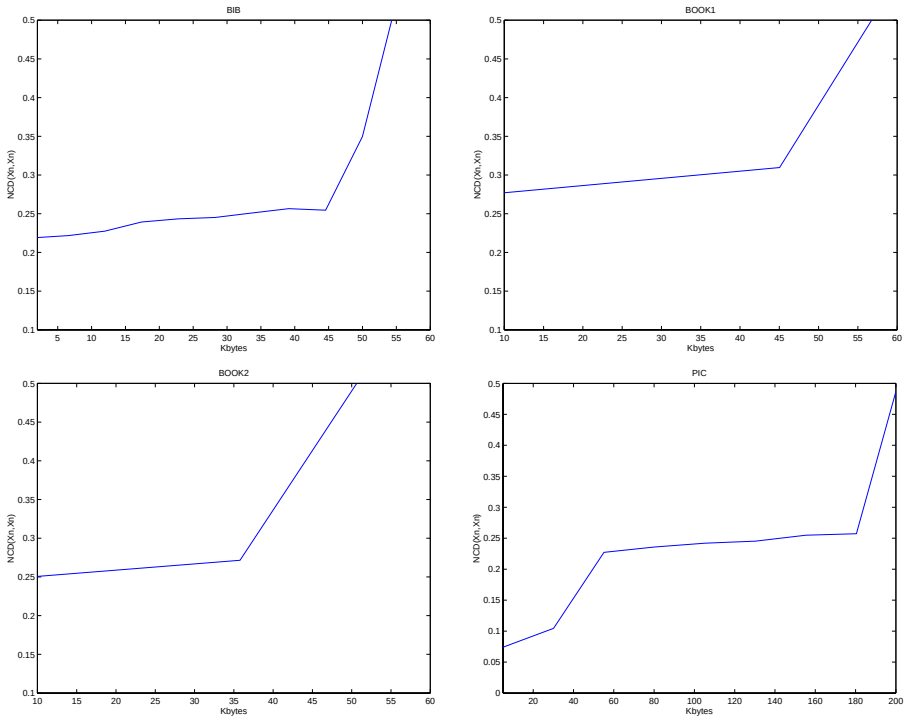

FIG. 8. *Normalized compression distances computed for the first n bytes of four files (bib, book1, book2 and pic, from left to right and top to bottom) of the Calgary Corpus files using the bzip2 compressor with the --fast option.*

the --fast option rather than the --best option (see Figure 8). In this case, the block size used by the compressor can be seen to be much smaller (about 100 Kbytes, vs. 900 in the --best case) which means that files over 50 Kbytes are not properly managed. Even the small files in our examples suffer now from this effect and show a strong dependency region.

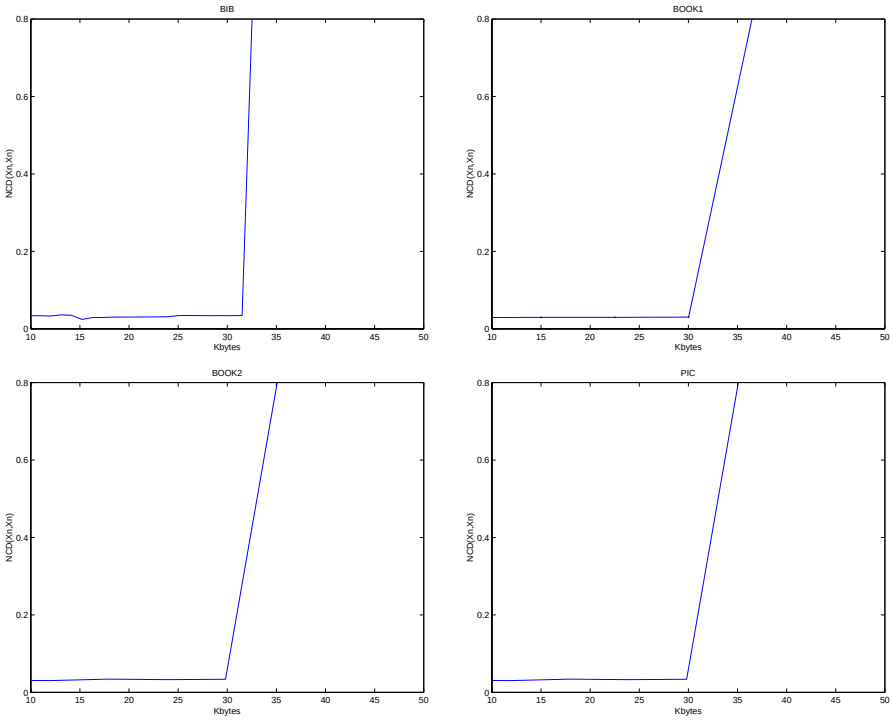

FIG. 9. *Normalized compression distances computed for the first n bytes of four files (bib, book1, book2 and pic, from left to right and top to bottom) of the Calgary Corpus files using the gzip compressor with the --best option.*

5.2. gzip. The plots in Figure 9, together with many more similar experiments we have performed, show that $\mathrm{NCD}(x, x)$ is between 0.0 and 0.1 in the region where gzip can be used properly, while it gives values which grow to 1 outside that region.

The experimental results obtained using the gzip compressor in the NCD are displayed in Figure 9. We can observe an initial slow-fluctuating growth with n, followed by a strong discontinuity, with a jump to 0.9 at 32 Kbytes, and finally a new slow (but slightly faster) growth, until the distance saturates in 1. We'll call again the two zones weak dependency and strong dependency, for analogy with the previous section.

The kernel of gzip [8] uses a variant of the LZ77 algorithm [9] for preprocessing and a statistical compressor (usually Huffman) as post-processing. The skew with the object size is fully explained by the compression scheme of the LZ77 algorithm[5].

As in the previous subsection, the two modes will be explained by means of three simple examples. The string to be compared with itself is again "drdobbs". This time there are two parameters that play the same role as the block size in the bzip2

[5]The Huffman coding does not have a relevant influence in the skew, so it is left out of this discussion.

current coding character	W_S	W_L	coded string
$\underset{\circ}{d}$rdobbsdrdobbs	empty	drbobbs	0_1
d$\underset{\circ}{r}$dobbsdrdobbs	d	rbobbsd	$0_1 0_1$
dr$\underset{\circ}{d}$obbsdrdobbs	dr	bobbsdr	$0_1 0_1 2_1$
drd$\underset{\circ}{o}$bbsdrdobbs	drd	obbsdrd	$0_1 0_1 2_1 0_1$
drdo$\underset{\circ}{b}$bsdrdobbs	drdo	bbsdrdo	$0_1 0_1 2_1 0_1 0_1$
drdob$\underset{\circ}{b}$sdrdobbs	drdob	bsdrdob	$0_1 0_1 2_1 0_1 0_1 1_1$
drdobb$\underset{\circ}{s}$drdobbs	drdobb	sdrdobb	$0_1 0_1 2_1 0_1 0_1 1_1 0_1$
drdobbs$\underset{\circ}{d}$rdobbs	drdobbs	drdobbs	$0_1 0_1 2_1 0_1 0_1 1_1 0_1 7_7$

FIG. 10. *W_S= 7 bytes, W_L = 7 bytes.*

compressor: the sliding window and the lookahead window. The sliding window W_S is a buffer that contains the previous $|W_S|$ characters to the character that is being compressed. On the other side, the lookahead window W_L is the buffer that contains the next $|W_L|$ characters that follow the character being coded. The LZ77 algorithm searches the longest string that begins in the current coding character and is contained in both windows.

In our first example, let's assume that $|W_S| = |W_L| = 7$ bytes. Remember that we want to compute $\frac{C(xx)-C(x)}{C(x)}$. The LZ77 algorithm is applied to the string "drdobbsdrdobbs". The sliding window and the lookahead window were large enough, and the compressor realized that the second half of the string is an exact repetition of the first. Let's assume that each compression chunk $\text{offset}_{\text{length}}$ has a size of 2 bytes (Figure 10). In this way $\frac{C(xx)-C(x)}{C(x)} = \frac{2}{14} = 0.143$. We can generalize: whatever the size of x, if the windows are large enough, $C(xx) - C(x) = 2$.

A new scenario is proposed: now $|W_S| = 7$ and $|W_L| = 3$ (see Figure 11). In this example, the compressor was unable to extract all the redundancy from the string, due to the insufficient size of the lookahead window. Rather than detecting that the second half of the string is identical to the first, the compressor only detects three substrings identical to three other substrings in the sliding window. This is what underlies the weak dependency. For the Calgary Corpus, the window size (32 Kbytes) is larger than the size of the object, but the lookahead window is smaller. This is why the NCD increases slightly with n in this zone: $C(xx) - C(x)$ is proportional to $\frac{|x|}{|W_L|}$. In our example, the distance has significantly increased: $\text{NCD} = \frac{20-14}{14} = 0.428$. This is a deviation in the distance, which depends little on the size of the objects.

It remains to explain the most important feature, the discontinuity point at 32 Kbytes. In this point, the size of the catenated object overflows the size of the sliding window.

current coding character	W_S	W_L	coded string
$\underset{\circ}{d}$rdobbsdrdobbs	empty	drd	0_1
d$\underset{\circ}{r}$dobbsdrdobbs	d	rdo	$0_1 0_1$
dr$\underset{\circ}{d}$obbsdrdobbs	dr	dob	$0_1 0_1 2_1$
drd$\underset{\circ}{o}$bbsdrdobbs	drd	obb	$0_1 0_1 2_1 0_1$
drdo$\underset{\circ}{b}$bsdrdobbs	drdo	bbs	$0_1 0_1 2_1 0_1 0_1$
drdob$\underset{\circ}{b}$sdrdobbs	drdob	bsd	$0_1 0_1 2_1 0_1 0_1 1_1$
drdobb$\underset{\circ}{s}$drdobbs	drdobb	sdr	$0_1 0_1 2_1 0_1 0_1 1_1 0_1$
drdobbs$\underset{\circ}{d}$rdobbs	drdobbs	drd	$0_1 0_1 2_1 0_1 0_1 1_1 0_1 7_3$
drdobbsdrd$\underset{\circ}{o}$bbs	obbsdrd	obb	$0_1 0_1 2_1 0_1 0_1 1_1 0_1 7_3 7_3$
drdobbsdrdobb$\underset{\circ}{s}$	sdrdobb	s	$0_1 0_1 2_1 0_1 0_1 1_1 0_1 7_3 7_3 7_1$

FIG. 11. *W_S= 7 bytes, W_L = 3 bytes.*

In our last example, we will assume that $|W_S| = 6$ and $|W_L| = 7$. The results are shown in Figure 12. The insufficient size of the sliding window causes the first byte of the string to be unreachable by the compressor, which loses all the redundancy detection. A very small change (only 1 byte) in the sliding window size can cause the compressor to be absolutely blind (the NCD calculation in this example gives $\frac{28-14}{14} = 1$. This is what causes the discontinuity at 32 Kbytes. When the size of the object is one byte more than the sliding window, the first byte of the first object is lost, and the compressor becomes unable to detect the full redundancy of the catenation, giving rise to an NCD value near to absolute dissimilarity (0.9 for almost all files in the Calgary Corpus).

The purpose of using the LZ77 algorithm in the NCD is based on the fact that it can use the sequences that appear in the first object to make the coding of the second object less expensive. If the size of the sliding window is significantly smaller than the size of any of the objects, the blind effect will outperform the redundancy detection task. From our experiments (those described in this paper and others) we can extract the following conclusion: if $|W_S| \ll |x|$ or $|W_S| \ll |y|$ then $\mathrm{NCD}(x, y) \approx 1$ for any possible value of $max\{K(x|y), K(y|x)\}/max\{K(x), K(y)\}$, i.e. for any similarity degree between x and y.

We have also repeated our experiments with gzip, by selecting the `--fast` option rather than the `--best` option (see Figure 13). In this case, the size of the sliding window used by the compressor does not change, so the results obtained are very similar to those with the `--best` option. Only the compress ratio obtained is affected (see Figure 15).

current coding character	W_S	W_L	coded string
$\overset{\circ}{d}$rdobbsdrdobbs	empty	drdobbs	0_1
d$\overset{\circ}{r}$dobbsdrdobbs	d	rdobbsd	$0_1 0_1$
dr$\overset{\circ}{d}$obbsdrdobbs	dr	dobbsdr	$0_1 0_1 2_1$
drd$\overset{\circ}{o}$bbsdrdobbs	drd	obbsdrd	$0_1 0_1 2_1 0_1$
drdo$\overset{\circ}{b}$bsdrdobbs	drdo	bbsdrdo	$0_1 0_1 2_1 0_1 0_1$
drdob$\overset{\circ}{b}$sdrdobbs	drdob	bsdrdob	$0_1 0_1 2_1 0_1 0_1 1_1$
drdobb$\overset{\circ}{s}$drdobbs	drdobb	sdrdobb	$0_1 0_1 2_1 0_1 0_1 1_1 0_1$
drdobbs$\overset{\circ}{d}$rdobbs	rdobbs	drdobbs	$0_1 0_1 2_1 0_1 0_1 1_1 0_1 5_1$
drdobbsd$\overset{\circ}{r}$dobbs	dobbsd	rdobbs	$0_1 0_1 2_1 0_1 0_1 1_1 0_1 5_1 0_1$
drdobbsdr$\overset{\circ}{d}$obbs	obbsdr	dobbs	$0_1 0_1 2_1 0_1 0_1 1_1 0_1 5_1 0_1 2_1$
drdobbsdrd$\overset{\circ}{o}$bbs	bbsdrd	obbs	$0_1 0_1 2_1 0_1 0_1 1_1 0_1 5_1 0_1 2_1 0_1$
drdobbsdrdo$\overset{\circ}{b}$bs	bsdrdo	bbs	$0_1 0_1 2_1 0_1 0_1 1_1 0_1 5_1 0_1 2_1 0_1 6_1$
drdobbsdrdob$\overset{\circ}{b}$s	sdrdob	bs	$0_1 0_1 2_1 0_1 0_1 1_1 0_1 5_1 0_1 2_1 0_1 6_1 1_1$
drdobbsdrdobb$\overset{\circ}{s}$	drdobb	s	$0_1 0_1 2_1 0_1 0_1 1_1 0_1 5_1 0_1 2_1 0_1 6_1 1_1 0_1$

FIG. 12. *W_S= 6 bytes, W_L= 7 bytes.*

6. Conclusions. In this paper, we have reviewed the concept of normalized compression distance (NCD), which uses compressors to provide a measure of the distance between two objects of any kind and can be used for clustering applications.

We have analyzed the impact on the NCD quality of some features of two compressors: the block size in bzip2, and the sizes of the two windows (sliding and lookahead) used by gzip. The well-known Calgary Corpus has been used as a benchmark. Any similarity distance should measure a 0 distance (or, at least, a very small value) between two identical objects. The empirical results obtained with both compressors for the Calgary Corpus reveal that the NCD is skewed by the size of the objects, independently of their type. For object sizes smaller than certain values (related to the block and window sizes in the compressors), the distance between two identical objects is usually quite small, which proves that the NCD is a good tool for this purpose. However, for larger sizes, when the inner limitations of the compressors are violated, obviously the distance between two identical objects grows to very high values, making the NCD practically unusable. Other widely used compressors (such as winzip and pkzip) also show the same limitations.

The use of block and window sizes in the compressors aims to increasing the computation speed at the expense of the compression ratio. This paper proves that

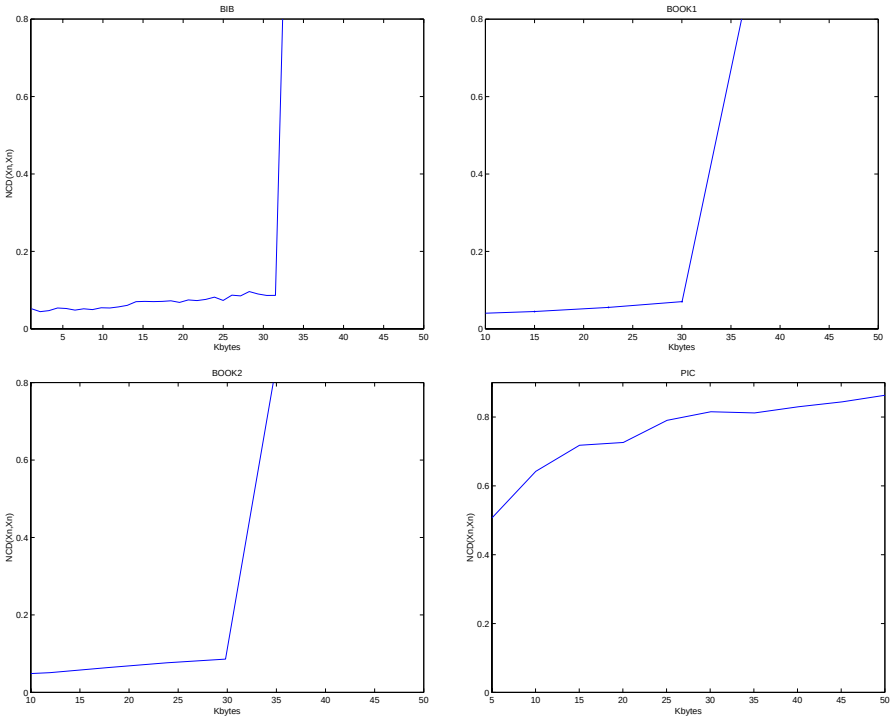

FIG. 13. *normalized compression distances computed for the first n bytes of four files (bib, book1, book2 and pic, from left to right and top to bottom) of the Calgary Corpus files using the gzip compressor with the --fast option.*

this balance between quality and speed should be treated carefully for clustering, where quality is tantamount. When considering clustering problems, all considerations about speed should be left apart if they imply exceeding the system parameters. The proper use of this powerful distance depends on selecting compression algorithms without limiting factors related to the size of the objects like, such as the high compression Markov predictive coder PPMZ [1], which does not set any window or block limit, but is much slower than those mentioned above. The results of using PPMZ in our experiments are shown in Figure 14 and are coherent with our conclusions: the distance computed with PPMZ does not depend on the size of the objects and is always between zero and a very small value (0.1043). On the other hand, this also confirms that the NCD is a very good distance measurement, when used in the proper way.

In the case of bzip2 and gzip, the block, the sliding window and the lookahead window should be at least as large as the sum of the sizes of the objects to be compared. The table in Figure 15 summarizes the results obtained for all three compressors under different circumstances, both as regards the compression ratio obtained and the size limits where the use of the NCD is acceptable for each.

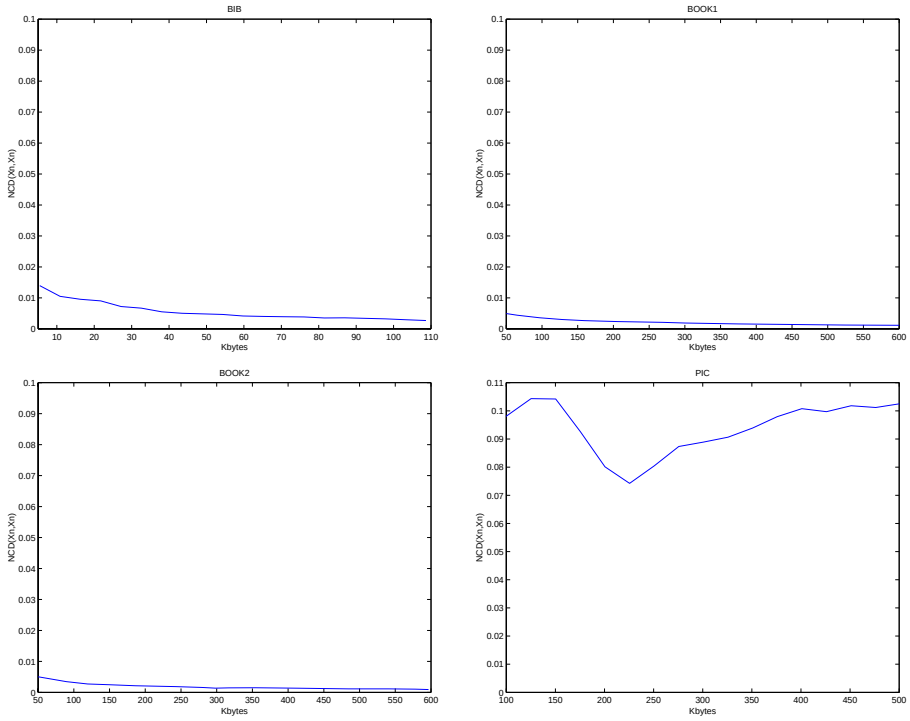

FIG. 14. *Normalized compression distances computed for the first n bytes of four files (bib, book1, book2 and pic, from left to right and top to bottom) of the Calgary Corpus files using the PPMZ compressor.*

compressor	options	comp. ratio	acceptable region	bad region
ppmz	none	25%	$(0, \infty)$	none
bzip2	--best	27%	$(0, 450]$	$(450 , \infty)$
bzip2	--fast	29%	$(0, 50]$	$(50, \infty)$
gzip	--best	32%	$(0, 32]$	$(32, \infty)$
gzip	--fast	38%	$(0, 32]$	$(32, \infty)$

FIG. 15. *Comparison table over the Calgary Corpus for all compressors and options used. The units of the acceptable and bad regions are in Kbytes. Both objects are the same size.*

Acknowledgements. This work was partially supported by grant TSI 2005-08255-C07-06 of the Spanish Ministry of Education and Science. We would also like to thank the reviewers for many useful comments.

REFERENCES

[1] J. CLEARY, I. WITTEN, AND A. C. CALGARY, *Data compression using adaptive coding and partial matching*, IEEE Trans. Communications, 32(1984), pp. 396–402.

[2] R. CILIBRASI AND P. M. B. VITÁNYI, *Clustering by compression*, IEEE Transactions on Infor-

mation Theory, 51:4(2005), pp. 1523–1545.

[3] M. Li, X. Chen, X. Li, B. Ma, and P. M. B. Vitányi, *The similarity metric*, IEEE Transactions on Information Theory, 50:12(2004), pp. 3250–3264.

[4] M. Li and P. B. M. Vitányi, *An Introduction to Kolmogorov Complexity and its Applications*, Springer-Verlag, New York, 2nd Edition, 1997.

[5] T. C. Bell, J. G. Cleary, and I. H. Witten, *Text compression*, Prentice Hall, Englewood Cliffs, New Jersey, 1990.

[6] M. Burrows and D. J. Wheeler, *A Block-sorting Lossless Data Compression Algorithm*, Systems Research Center of Digital Equipment Corporation, Technical Report 124, Palo Alto, California, 1994.

[7] D. Huffman, *A Method for the Construction of Minimum Redundancy Codes*, Proc. IRE, Vol. 40, No. 9, 1952.

[8] DEFLATE Compressed Data Format Specification available at ftp://ds.internic.net/rfc/rfc1951.txt

[9] J. Ziv and A. Lempel, *A universal algorithm for sequential data compression*, IEEE Transactions on Information Theory, 23:3(1997), pp. 337–343.