Various Compression-based Similarity Measures that have been proposed and applied to Machine Learning related Tasks

Professor Ja-Ling Wu
Dept. of Computer Science and Information Engineering
Graduate Institute of Networking and Multimedia

1

The following discussions are cited from the work:
"**Compression and Machine Learning: A New Perspective on Feature Space Vectors**",
written by:
D. Sculley and Carla E. Brodley

2

The fundamental idea that data compression can be used to perform machine learning tasks has surfaced in a several areas of research, including

data compression (Witten et al., 1999a; Frank et al., 2000), **m learning-based data mining** (Cilibrasi and Vitanyi, 2005; Keogh et al., 2004; Chen et al., 2004),
information theory, (Li et al., 2004),
bioinformatics (Chen et al., 1999; Hagenauer et al., 2004),
spam filtering (Bratko and Filipic, 2005), and even
physics (Benedetto et al., 2002).

3

The principle at work is that **if strings x and y compress more effectively together than they do apart**, then they must **share similar information**.

工作原理是，如果字符串 **x** 和 **y**合在一起壓縮比分開壓縮更有效率，那麼它們必定共享更多相似的信息。

4

Compression-Based Similarity Measures

We will use **|x|** to denote **the number of symbols in string x**.

C(x) gives the **length of string x** after it has been compressed by **compression algorithm C(·)** measured as a **number of bits** (most often rounded up to whole bytes).

The **concatenation of strings x and y** is written as **xy**; thus, **C(xy)** gives the number of bits needed to **compress x and y together**.

5

The term **C(x|y)** shows the **length of x when conditionally compressed with a model built on string y**.

Some researchers have implemented specialized conditional compression algorithms (Chen et al., 1999), the approximation C(x|y) = C(yx) − C(y) accommodates the use of off-the-shelf compressors (Li et al., 2004).

6

NCD: the Normalized Compression Distance.
The **NCD** is defined as follows:

$$NCD(x,y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}$$

This metric has the advantage of "**minorizing in an appropriate sense every effective metric**", which means that when NCD says that two strings are strongly related, they very likely are. Cilibrasi and Vitanyi (2005) show that **NCD is a formal distance metric within certain tolerances by defining bounds** on what they called a normal compressor, which include the requirement that C(x) = C(xx) within logarithmic bounds.

7

When these bounds are met, **NCD** operates in the range **[0, 1+ϵ]**, where **0 shows x and y are identical**, and **1 shows they are completely dissimilar**. Here, ϵ is a small **error term** on the order of **.1** (Li et al., 2004).

Although some standard compression algorithms such as **LZ77**, **LZ78**, and even **PPM**, are **not guaranteed to satisfy these bounds**, **NCD** has been successfully applied to a host of **clustering applications** (Cilibrasi and Vitanyi, 2005; Cilibrasi et al., 2004; Li et al., 2004).

8

CLM: the Chen-Li Metric.
The CLM is formulates as:

$$CLM(x,y) = 1 - \frac{C(x) - C(x|y)}{C(xy)}$$

The term **C(x) − C(x|y) gives an upper bound on mutual algorithmic information between strings**, a measure of shared information.
Like NCD, this metric is normalized to the range **[0,1]**, with **0 showing complete similarity and 1 showing complete dissimilarity**. **CLM** has achieved **empirical success in several important applications**, including **genomic sequence comparison** and **plagiarism detection** (Chen et al., 1999; Li et al., 2001; Chen et al., 2004).

9

CDM: the **Compression-based Dissimilarity Measure**. Keogh et al. (2004) set forth their CDM in response to NCD, calling it a "simpler measure," but avoiding theoretical analysis.

$$CDM(x,y) = \frac{C(xy)}{C(x) + C(y)}$$

These authors are aware that **CDM is non-metric, failing the identity property**. CDM gives values in the range of **[1/2,1], where 1/2 shows pure identity and 1 shows pure disparity**. But although CDM was proposed without theoretical analysis, it was used to produce successful results in **clustering** and **anomaly detection** (Keogh et al., 2004).

10

CosS: Compression-based Cosine. The compression-based measure, CosS, is defined based on the familiar cosine-vector dissimilarity measure:

$$CosS(x,y) = 1 - \frac{C(x) + C(y) - C(xy)}{\sqrt{C(x)C(y)}}$$

This measure is normalized to the range **[0,1]**, with **0 showing total identity between the strings, and 1 total dissimilarity**.

Although the formulae of each of these four measures appears to be quite distinct from the others, we will show in Section 3.2 that the only actual differences among them are in the **normalizing terms**.

11

Compression and Feature Vectors

First, for each compressor C(·) we define an associated **vector space ℵ ,** such that **C(·) maps an input string x into a vector $\vec{x}$ ∈ℵ** . Second, we must show that **the value C(x), the length of the compressed string x,** corresponds to a **vector norm $||\vec{x}||$.**
We examine three representative lossless compression methods, **LZW, LZ77, and PPM**. Later, we will discuss the connection between **string concatenation** and **vector addition**, as the addition operator will be useful when we take apart the formulae of the similarity measures to examine their effect in the implied feature spaces.

12

LZ77 Feature Space.

The LZ77 algorithm, prototypical of one branch of the **Lempel-Ziv** family of compressors, **encodes substrings as a series of output codes** that **reference previously occurring substrings in a sliding dictionary window**. Although parameter values vary by implementation, we may assume that **the repeated substrings found by LZ77 are of maximum length m**, and that **the dictionary window is of length p ≥ m**. Furthermore, we will for simplicity assume an implementation of LZ77 in which **the output codes are of constant length c**.

13

The **feature space** $\aleph$, is a **high dimensional space** in which **each coordinate corresponds to one of the possible substrings of up to length m**.

The **compressor** implicitly **maps string x to $\vec{x} \in \aleph$** as follows.

Initially, each element $\vec{x}_i = 0$. As LZ77 passes over the string and produces output codes, **each new output code referring to substring i causes the update $\vec{x}_i := \vec{x}_i + c$. At the end of the process, the implicit $\vec{x}$ is a vector modeling x**.

14

Furthermore, as **all $\vec{x}_i$ are non-negative**, C(x) yields the **1-norm of $\vec{x}$**, written $\|\vec{x}\|_1 = \sum_i |\vec{x}_i|$, which is sometimes referred to as the **city block distance.**

Note that although the LZ77 feature space ℵ is very large, with **$O(2^m)$ dimensions**, mapping from x to $\vec{x} \in \aleph$ is **fast,** taking only **$O(|x|)$ operations**.

15

LZW Feature Space.

The other branch of the Lempel-Ziv family is represented by the LZW algorithm, which is closely related to **LZ78** and a host of variants.

Unlike LZ77, which refers to strings in a **sliding dictionary window**, **LZW builds and stores an explicit substring dictionary on the fly**, **selecting new substrings to add to the dictionary using a greedy selection heuristic**.

Output codes refer to substrings already in the dictionary; we will again assume **fixed-length output codes of size c** for simplicity, and assume that the **dictionary can hold $O(2^c)$ entries**.

16

With the **standard LZW substring selection heuristic**, the **maximum length of a substring** in the **LZW dictionary** is also **$O(2^c)$.** The **vector space ℵ implied by LZW is thus actually larger than that of LZ77** (assuming reasonable parameter values for each), and has **one dimension for each possible substring with maximum length $O(2^c)$,** that is, **$O(2\uparrow(2^c))$ dimensions**. LZW implicitly maps x to $\vec{x} \in$ ℵ as follows; each $\vec{x}_i$ is **initialized at zero**, and is **incremented by c when an output code corresponding to dimension i is produced**.

Despite the high dimensionality of ℵ , the **mapping is completed in time $O(|x|)$,** and at the **end** of the process, **$C(x) = ||\vec{x}||_1$**.

17

PPM Feature Space.

Extremely effective lossless compression is achieved with **arithmetic encoding under a model of prediction by partial matching** (Witten et al., 1999b).

Arithmetic coding is a compression technique that attempts to make C(x) **as close as possible to the ideal − log P (x) by allowing symbols within a message to be encoded with fractional quantities of bits**. An **order-n predictive model** is generated by **building statistics for symbol frequencies**, based on a **Markovian assumption** that the **probability distribution of symbols at a given point relies only on the previous n symbols in the stream**.

18

At each step, **PPM encodes a symbol s following the n-symbol context t**, using **$c \approx -\log P(s|t)$ bits**. Because of arithmetic coding, **c is positive**, but is **not necessarily a whole number**.

Note that the **probability estimate P(s|t) is the algorithm's estimate at that step**, and **these estimates change during compression as the algorithm adapts to the data**.

However, the details of the probability estimation scheme, which differ by implementation and algorithm variant, **do not alter the implicit feature space**.

19

The implicit **PPM feature space** $\aleph$ has **one coordinate for each possible combination of symbol s and context t,** which may **be thought of together as a single string ts.**
Thus, an **order-n PPM feature space** has **one dimension for each possible string of length n + 1**. **PPM maps x to $\vec{x} \in \aleph$** by beginning with **each $\vec{x}_i = 0$**.

For each **new symbol-context pair ts** encountered during compression, **where s is encoded with c bits** and **ts corresponds to dimension i in $\aleph$, $\vec{x}_i := \vec{x}_i + c$**.

At the end of compression, then, **$C(x) = ||\vec{x}||_1$**.

20

String Concatenation and Vector Addition.

We have just shown that **each of the above-mentioned prototypical compression algorithms has an associated feature space, ℵ** , that each compressor maps a string **x into a vector $\vec{x} \in$ ℵ** and that for each compressor, **$C(x) = \|\vec{x}\|_1$**.
This is the foundation of the analysis of compression-based similarity measures in vector space. But before we can take apart the similarity measures, themselves, we need to examine **the effect of string concatenation**.

21

We define **concatenation as string addition**: strings **$x + y = xy$**. **Compressing xy maps each string into a vector and adds the vectors together**. Thus, **$C(xy) = \|\vec{x} + \vec{y}\|_1$**, which satisfies the **triangle inequality requirement** of vector norms: $\|\vec{x}+\vec{y}\| \leq \|\vec{x}\|+\|\vec{y}\|$, because $C(xy) \leq C(x)+C(y)$ (in the absence of pathological cases; see below.) Note that **the salient quality of the compressors here is that they are adaptive**. In the absence of adaptive compression, $C(x) + C(y) = C(xy)$ for all strings, and **machine learning is impossible**. **With adaptive compression**, **$\max\{C(x),C(y)\} \leq C(xy) \leq C(x) + C(y)$**, just as **$\max\{\|\vec{x}\|_1, \|\vec{y}\|_1\} \leq \|\vec{x} + \vec{y}\|_1 \leq \|\vec{x}\|_1 + \|\vec{y}\|_1$** if all elements of $\vec{x}$ and $\vec{y}$ are **non-negative.**
Adaptive compression of concatenated strings performs vector addition within the implicitly defined feature space.

22

Yet a few caveats (警告) are in order. First, **the commutative property of addition is not strictly met with all compressors.** Because of string alignment issues and other details such as model flushing, many adaptive compression algorithms are **not purely symmetric** – that is, **C(xy) is not exactly equal to C(yx).** Second, in some cases, even **the triangle inequality requirement may fail to hold**. If the initial string x uses up the entire dictionary space in Lempel-Ziv methods, **C(xy) > C(x)+C(y).** And under **PPM**, if **x is very different from y, it is possible that C(xy) > C(x)+C(y),** depending on the nature of the adaptive modeling scheme.
In this case, the presence of model flushing is a benefit, keeping C(xy) close to C(x) + C(y) when x and y are highly dissimilar.

23

Measures in Vector Space

Measuring distance between our implicit vectors would be simple given a subtraction operator defined for the implicit vectors formed by compression mapping. If subtraction were available, we could use a quantity like **$||\vec{x}-\vec{y}||$ as a distance metric**. However, **the only available vector operators in this implicit space are addition of vector magnitudes**.
It turns out that all four of the compression-based similarity measures address this issue in the same way, and use the quantity **$||\vec{x}||_1 + ||\vec{y}||_1 - ||\vec{x}+\vec{y}||_1$** as a vector similarity measure.

24

Simple transformations show that this term $(||\vec{x}||_1 + ||\vec{y}||_1 - ||\vec{x}+\vec{y}||_1)$ occurs in each compression-based measure. When **x and y are very similar**, the term approaches $\max\{||\vec{x}||_1, ||\vec{y}||_1\}$. When **the two share little similarity, the term approaches 0**.

However, when left **un-normalized**, the term **may show more absolute similarity between longer strings than shorter strings**. To allow meaningful comparisons in similarity between **strings of different lengths**, the measures are **normalized**. As shown in Table 1, all of the measures can be reduced to a **canonical form** $1 - \{\{||\vec{x}||_1 + ||\vec{y}||_1 - ||\vec{x}+\vec{y}||_1\} / f(\vec{x}, \vec{y})\}$ where $f(\vec{x}, \vec{y})$ is a particular normalizing term. (Subtracting the normalized similarity term from 1 makes these measures dissimilarity measures.)

25

Thus, the only differences among the measures is in the choice of **normalizing term**:

CosS normalizes by the **geometric mean** of the two vector magnitudes,
CDM by **twice the arithmetic mean**,
NCD by **the max of the two**,
CLM by **the magnitude of the vector sum**.

Indeed, in the experimental section, we see that the four measures give strikingly similar results.

26

$CosS$	$= 1 - \frac{C(x)+C(y)-C(xy)}{\sqrt{C(x)C(y)}} = 1 - \frac{\lvert\lvert\vec{x}\rvert\rvert_1+\lvert\lvert\vec{y}\rvert\rvert_1-\lvert\lvert\vec{x}+\vec{y}\rvert\rvert_1}{\sqrt{\lvert\lvert\vec{x}\rvert\rvert_1\lvert\lvert\vec{y}\rvert\rvert_1}}$
CLM	$= 1 - \frac{C(x)-C(x\vert y)}{C(xy)} = 1 - \frac{C(x)+C(y)-C(xy)}{C(xy)} = 1 - \frac{\lvert\lvert\vec{x}\rvert\rvert_1+\lvert\lvert\vec{y}\rvert\rvert_1-\lvert\lvert\vec{x}+\vec{y}\rvert\rvert_1}{\lvert\lvert\vec{x}+\vec{y}\rvert\rvert_1}$
CDM	$= \frac{C(xy)}{C(x)+C(y)} = 1 - \left(1 - \frac{C(xy)}{C(x)+C(y)}\right) = 1 - \left(\frac{C(x)+C(y)}{C(x)+C(y)} - \frac{C(xy)}{C(x)+C(y)}\right)$ $= 1 - \frac{C(x)+C(y)-C(xy)}{C(x)+C(y))} = 1 - \frac{\lvert\lvert\vec{x}\rvert\rvert_1+\lvert\lvert\vec{y}\rvert\rvert_1-\lvert\lvert\vec{x}+\vec{y}\rvert\rvert_1}{\lvert\lvert\vec{x}\rvert\rvert_1+\lvert\lvert\vec{y}\rvert\rvert_1}$
NCD	$= \frac{C(xy)-\min\{C(x),C(y)\}}{\max\{C(x),C(y)\}} = 1 - \left(1 - \frac{C(xy)-\min\{C(x),C(y)\}}{\max\{C(x),C(y)\}}\right)$ $= 1 - \left(\frac{\max\{C(x),C(y)\}}{\max\{C(x),C(y)\}} - \frac{C(xy)-\min\{C(x),C(y)\}}{\max\{C(x),C(y)\}}\right)$ $= 1 - \frac{C(x)+C(y)-C(xy)}{\max\{C(x),C(y)\}} = 1 - \frac{\lvert\lvert\vec{x}\rvert\rvert_1+\lvert\lvert\vec{y}\rvert\rvert_1-\lvert\lvert\vec{x}+\vec{y}\rvert\rvert_1}{\max\{\lvert\lvert\vec{x}\rvert\rvert_1,\lvert\lvert\vec{y}\rvert\rvert_1\}}$

Table 1: Reducing similarity measures to canonical form.

27

From Feature Vectors to Compression

We have shown that **compression maps strings into feature vectors**, and have looked briefly at the similarity measures applied in the feature spaces.

Now we show that **standard explicit feature vectors (which were used to solve mining and classification problems) have strong links back to compression**. These connections show the potential for improved explicit models based on insights from adaptive compression methods.

28

TF*IDF.

One conventional approach to **text representation** is the TF*IDF vector model (Salton and Buckley, 1988), in which **coordinates of the vector space correspond to individual words**, and are given a **score based on the term frequency times its inverse document frequency**.

The value of each $\vec{x}_i$ corresponding to **a word w_i** **occurring n_i times** in **string x containing |x| total words**, and which occurs with **probability P (w_i) in some reference corpus**, is given by $\vec{x}_i = n_i \log 1/P(w_i)$.

29

The connection to compression is clear: a word-based compression algorithm, given a fixed probability distribution P(w) for all possible words, will **compress x to a length C(x)** such that:

$$\sum_i n_i \log \frac{1}{P(w_i)} = ||\vec{x}||_1 = C(x)$$

Thus, the only difference between a **word-based compression method** and **TF*IDF** is that the latter **represents its feature space explicitly** – which is an advantage for certain learning algorithms such as **SVM** or **decision trees**.

30

Yet the insight begs the question:

if **compression algorithms** are able to **better model** the data by **adapting their estimation of the probability distribution P(w)** during compression, may **TF*IDF** achieve better results using an **adaptive scoring method** inspired by compression techniques?

31

Binary Bag of Words.

The binary bag of words method **gives the elements of a word vector a binary {0,1} score indicating presence or non-presence of a word**, but **ignores the number of repetitions**.

This may be viewed as equivalent to a form of **lossy compression of a given text**, in which **all words are given equal probability**, but the frequencies of words in the text are discarded in compression.

32

n-Grams and k-Mers.

In the **n-gram feature model**, the vector space has **one coordinate for each possible substring of n symbols**, called an **n-gram** (or, alternately, a **k-mer** or **p-spectrum**) (Shawe-Taylor and Cristianini, 2004).

The **score** for an **element x_i** of an n-gram vector representing **string x is the count of n_i times** that the (**possibly overlapping**) **n-gram g_i appears in x**.

33

The link to compression is made with a **uniform probability distribution P(g)**:

$$\sum_i n_i \log \frac{1}{P(g_i)} = ||\vec{x}||_1 = C(x)$$

With compression techniques at hand, we can see the potential for an n-gram method with adaptive probability distributions inspired by the **PPM algorithm** as an area of great interest.

34

An Empirical Test

As an initial confirmation of the tight connection between compression-based similarity methods and explicit feature vector models, the authors conducted a classification experiment on the **Unix User Data Set** archived at the UCI machine learning repository (Blake and Merz, 1998). They selected this publicly available data set, as it is **free from some of the ambiguities and noise** that occur in other benchmark data sets.

This makes classification by the **Nearest Neighbor technique** a fair test, allowing direct comparison of similarity measures drawn from various data models.

35

The UNIX user data set contains **labeled transcripts of nine Unix system users**, developed for **testing intrusion detection** (入侵檢測) systems (Lane and Brodley, 2003). This data is applied to a **user classification problem**: **given a test string x of Unix commands and a training set of labeled user sessions, identify the user who generated x.**

The authors employed the **Nearest Neighbor** classification method, first using the four compression-based similarity measures NCD, CDM, CLM, and CosS in combination with each of the compression algorithms discussed above.

36

The tests are then **repeated with three standard explicit feature vector models**, the **binary bag of words, TF*IDF vectors, and n-gram models**, using the established **cosine-vector similarity measure** to compute similarity scores.

Tests were run for **1000 randomly selected test sessions**, with **a minimum length of 10 Unix commands** in the session string. The **accuracy** is reported as the measure of success for each method, based on **the number of correct Nearest Neighbor classifications**.

37

The results, detailed in Table 2, show that **the compression-based methods compete with, and even exceed, the performance of common explicit vector space methods**.

It is interesting to note that the **PPM methods out-perform the associated n-gram methods**, despite the two **sharing the same feature space**. This suggests that the **weights** implicitly **assigned to the coordinates in the vector space by PPM** were indeed **informative**.

38

Compressor	NCD	CosS	CDM	CLM
PPM 3	.801	.834	.836	.839
PPM 4	.808	.828	.830	.830
LZ77	.735	.710	.725	.720
LZW	.669	.691	.691	.714
Vector Model	BINARY BAG	TF*IDF	4-GRAM	5-GRAM
	.838	.791	.777	.759

Table 2: Unix User ID Results. Accuracy is reported over 1000 trials.

39

Concluding Remarks:

Perhaps the most difficult problem in machine learning and data mining is **choosing an appropriate representation for the data**. At first blush, compression-based methods seem to side-step (迴避) this problem, but more thorough examination shows that **a choice of compression algorithm implies a specific, definable representation of the data within a feature space**.

This view allows compression-based similarity measures to be justified on the same terms as **explicit feature vector models** such as **TF*IDF** and **n-gram models**.

40

However, there are **a number of limitations on the use of standard compression algorithms for machine learning**.

First, the **No Free Lunch Theorem** (Wolpert and Macready, 1995) told us that **there can be no algorithm that losslessly compresses all possible strings**, just as there is **no machine learning method that automatically learns from all data**.

The choice of compression algorithm implies a particular set of features, and these must align well with the chosen data.

41

Second, computing similarity with **off-the-shelf compression algorithms** may require **more computational overhead** than using **explicit feature vector methods**. While both forms of similarity may be computed in time $O(|x| + |y|)$, in practice the **constant for compression algorithms is much greater**.

Third, and perhaps most importantly as demonstrated by Frank et al. (2000), **explicit feature models are more easily used by the full range of machine learning algorithms**.

42

References

D. Benedetto, E. Caglioti, and V. Loreto. Language trees and zipping. Phys. Review Lett., 88(4), 2002.
C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. URL http://www.ics.uci.edu/~mlearn/MLRepository.html.
A. Bratko and B. Filipic. Spam filtering using compression models. Technical Report IJS-DP-9227, Department of Intelligent Systems, Jozef Stefan Institute, Ljubljana, Slovenia, 2005.
X. Chen, B. Francia, M. Li, B. Mckinnon, and A. Seker. Shared information and program plagiarism detection. IEEE Trans. Information Theory, 7:1545–1550, 2004.
X. Chen, S. Kwong, and M. Li. A compression algorithm for DNA sequences and its applications in genome comparison. In Genome Informatics: Proceedings of the 10th Workshop on Genome Informatics, pages 51–61, 1999.
R. Cilibrasi and P. Vitanyi. Clustering by compression. IEEE Trans. Information Theory, 51:4, 2005.
R. Cilibrasi, P. Vitanyi, and R. de Wolf. Algorithmic clustering of music based on string compression. Computer Music Journal, 28:49–67, 2004.

43

J. Cleary and I. Witten. Data compression using adaptive coding and partial string matching. IEEE Transactions on Communcations, 32(4):396–402, 1984.
E. Frank, C. Chui, and I. H. Witten. Text categorization using compression models. In Proceedings of DCC-00, IEEE Data Compression Conference, pages 200–209. IEEE Computer Society Press, Los Alamitos, US, 2000.
Marcus Greelnard. Basic Compression Library, 2004. URL bcl.sourceforge.net.
P. Gru ̈nwald. A tutorial introduction to the minimal description length principle. In P. Gru ̈nwald, I. J. Myung, and M. Pitt editors, Advances in Minimum Description Length: Theory and Appli- cations, MIT Press, 2005.
J. Hagenauer, Z. Dawy, B. Goebel, P. Hanus, and J. Mueller. Genomic analysis using methods from information theory. IEEE Information Theory Workshop (ITW), pages 55–59, October 2004.
D. Hankerson, G. A. Harris, and P. D. Johnson. Introduction to Information Theory and Data Compression, 2nd ed. Champan and Hall, 2003.
E. Keogh, S Lonardi, and C. Ratanamahatana. Toward parameter-free data mining. ACM SIGKDD, pages 206–215, 2004.

44

D. V. Khmelev and W. J. Teahan. A repetition based measure for verification of text collections and for text categorization. In SIGIR ’03: Proceedings of the 26th Annual International ACM SIGIR Conference, pages 104–110. ACM Press, 2003.
T. Lane and C. E. Brodley. An empirical study of two approaches to sequence learning for anomaly detection. Machine Learning, 51(1):73–107, 2003.
M. Li, J. H. Badger, X. Chen, S. Kwong, P. Kearney, and H. Zhang. An information-based sequence distance and its application to whole mitochondrial genome phylogeny. Bioinformatics, 17(2): 149–154, 2001.
M. Li, X. Chen, X. Li, B. Ma, and P. Vitanyi. The similarity metric. In Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms, pages 863–872, 2004.
M. Li and P. Vitanyi. An Introduction to Kolmogorov Complexity and its Application. 2nd ed. Springer, 1997.
M. Nelson. LZW data compression. Dr. Dobb’s Journal, pages 29–36, October 1989.
M. Nelson. Arithmetic coding and statistical modeling. Dr. Dobbs Journal, pages 16–29, February 1991.

45

G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. Inf. Process. Manage., 24(5):513–523, 1988.
K. Sayood. Introduction to Data Compression, 2nd ed. Morgan Kaufmann, 2000.
J. Shawe-Taylor and N. Cristianini. Kernel Methods for Pattern Analysis. Cambridge University Press, 2004.
L. N. Trefethen and D. Bau. Numerical Linear Algebra. Society for Industrial and Applied Mathematics, 1997.
T. Welch. A technique for high-performance data compression. Computer, pages 8–19, June 1984. I. H. Witten, Z. Bray, M. Mahoui, and W. J. Teahan. Text mining: A new frontier for lossless compression. In Data Compression Conference, pages 198–207, 1999a.
I. H. Witten, A. Moffat, and T. C. Bell. Managing Gigabytes: Compressing and Indexing Documents and Images, 2nd ed. Morgan Kaufmann, 1999b.
D. H. Wolpert and W.G. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute, Santa Fe, NM, USA, 1995.
J. Ziv and A. Lempel. A universal algorithm for sequential data compression. IEEE Transactions on Information Theory, 23(3):337–343, 1977.

46