

Capítulo 1. Teoría de la información.

Por Ángel Fernando Kuri Morales

La teoría de la información es una disciplina que se centra en el enfoque matemático al estudio de la recopilación y manipulación de información. La información puede estar contenida, básicamente, en cualquier tipo de objeto que sea representable en la computadora, por ejemplo en un texto, una pintura, un sonido, una película, etc. El concepto de información por sí mismo es interesante y hay que recalcar que, en el presente capítulo, se dan dos definiciones precisas de tal forma que podamos hablar de información independientemente del contexto personal o cultural en el que el objeto de nuestro estudio se encuentre enclavado.

1.1. Información y significado.

Una de las preguntas que surgen de manera natural cuando se lee un texto es “¿Qué tan útil es lo que estoy leyendo?”. Es una pregunta lógica cuando se desea determinar si el esfuerzo implicado en la lectura nos dará algún beneficio. La misma pregunta podría hacerse al observar una pintura o una película, o al escuchar una pieza musical. Esta pregunta, sin embargo, es muy difícil de responder en primera instancia. Por ejemplo, podríamos preguntarnos el impacto que el conocimiento derivado de la lectura del texto tendrá en nuestra vida diaria. En ese caso habría que cuantificar en primer lugar el conocimiento y en segundo lugar el beneficio obtenido. Ambas cosas (el conocimiento y el beneficio) son eminentemente subjetivas. Podríamos, entonces, tratar de ser más objetivos y determinar la información contenida en el objeto de nuestro análisis para, posteriormente, determinar el valor intrínseco de la información en él contenida. Por ello es que analizar la información para saber qué es y cómo se mide tiene una importancia fundamental en la sociedad actual, que utiliza tantos instrumentos que almacenan y transmiten toda suerte de datos.

El tratamiento actual de la teoría de la información tiene dos vertientes, la teoría estadística de la información y la teoría algorítmica de la información. Ambas miden lo mismo, pero la definen de diferentes maneras. Aquí haremos una breve descripción de ellas y las contrastaremos. Sin embargo, antes de pasar a su estudio y al de sus posibles aplicaciones, vale la pena hacer una breve digresión para tratar de hacer más intuitivo su objeto.

Supongamos que a un lector que se mantiene más o menos al tanto de los resultados en el deporte más popular del mundo y que no es un experto en computación se le presentan los siguientes dos oraciones:

- a) El gol que anotó el centro delantero definió el resultado de la copa mundial.
- b) Un algoritmo genético canónico se puede modelar como una cadena de Markov ergódica.

Si a este lector típico le preguntamos cuál de las dos frases tiene mayor información, lo más probable es que indique que es la primera, pues le da más información que la segunda. Más aún, posiblemente indicará que la segunda oración no conlleva ninguna información, ya que no es capaz de encontrarle sentido. Esta es una actitud típica y explicable. Supongamos ahora que el lector es, además, hispano parlante y no bilingüe. A nuestro atribulado lector le daría lo mismo que le hubiéramos entregado, en vez de la segunda oración mostrada arriba, la siguiente oración:

- c) The goal scored by the center forward defined the outcome of the World Cup.

Aquí se manifiesta el hecho de que ni la oración (b) ni la (c) tienen significado para un lector con las características indicadas. Y es aquí en donde tenemos que enfatizar la diferencia que existe entre significado e información. Si el lector de este libro no habla inglés, debe saber que la oración (c) dice exactamente lo mismo que la (a), solamente que en inglés. Aunque, al no ser anglo parlantes, seamos incapaces de extraer significado de (c), debe resultar evidente (ahora que se nos ha dicho que significan lo mismo) que tanto (a) como (c) contienen la misma

información. La moraleja que queremos extraer de este sencillo ejemplo es que no hay que confundir, como frecuentemente ocurre, significado con información. El **significado** es algo que manifiesta un isomorfismo (formas iguales) entre “viejas” cosas conocidas y nuevas cosas desconocidas que se presentan a nuestra atención. La oración (b), por ejemplo, nos resulta probablemente ininteligible porque muchos de los términos contenidos en ella, como “algoritmo”, “genético”, “canónico”, “cadena de Markov”, y “ergódica”, son términos especializados que probablemente nos son desconocidos. Es decir, no podemos establecer una relación entre esos términos **que no nos son conocidos** y otros que sí lo son; no podemos establecer isomorfismos. En cambio, los términos usados en (a), como “gol”, “anotó”, “centro delantero” y “copa mundial”, seguramente sí son más conocidos. No solamente eso, sino que de manera tácita se encuentran isomorfismos que no se hacen explícitos en (a). Por ejemplo, aunque no se diga en ninguna parte, el lector de nuestro ejemplo probablemente entiende (“sabe”) que la “copa” a la que nos referimos es, en realidad, un torneo y que la oración se refiere al deporte llamado fútbol soccer. En realidad la oración (a) invoca información que ya se encuentra en el sujeto que trata de interpretarla.

Un ejemplo extremo de lo anteriormente expuesto se puede encontrar en un movimiento artístico y filosófico denominado “Fluxus”. Para los exponentes de este movimiento, el público debe formar parte de la obra de arte. Por ejemplo, el autor musical John Cage desarrolló una obra musical para piano llamada 4’33 (Wikipedia, 2006). Esta obra consiste de 4 minutos y 33 segundos de...silencio. Para Cage (y otros convencidos de la idea de Fluxus), sin embargo, la expectación de ver a un ejecutante que no tocaba el piano a su disposición, sino que esperaba, intencionalmente, durante unos momentos en silencio, cambiando las hojas de una partitura sin tocar el teclado, involucraba al espectador y lo hacía parte de la creación artística. En esta obra hay, sin que los espectadores (y tal vez ni el mismo Cage) hagan conciencia, un

sin número de isomorfismos con otras obras musicales, con el ritual del ejecutante, con las expectativas del público, con la sala de conciertos, con el ruido de fondo, etc. Para los entusiastas de Fluxus, esta obra está plena de significado. Para nosotros, sin embargo, carece de información.

1.2. La teoría de la información.

Al objeto bajo análisis codificado en nuestra computadora le llamaremos, de aquí en adelante, el **mensaje**. Todo mensaje requiere de un sujeto que lo origina (la **fuentes**) y otro que lo recibe (el **receptor**). No tiene sentido práctico hablar de un mensaje si éste no se transmite. Al medio a través del cual se transmite el mensaje se le denomina el **canal de transmisión** (o simplemente, el **canal**). Para efectos de nuestra teoría no tiene ninguna importancia el medio físico del que está constituido dicho canal. De hecho, el canal puede no existir en el sentido físico. Por ejemplo, si nosotros almacenamos un conjunto de datos hoy para extraerlo mañana, el mensaje no se transmite a través del espacio sino a través del tiempo. Pero el mensaje (los datos) ciertamente se origina (se escribe hoy), se transmite (se almacena para que persista en el tiempo) y se recibe (se lee mañana).

Dos autores se distinguieron por su planteamiento matemático que aborda el problema de la información contenida en un mensaje. El primero de ellos, Claude Shannon (Shannon, 1948), abordó el problema desde un punto de vista estadístico. El segundo, Andrei Kolmogorov (Kolmogorov, 1965)¹ lo hizo desde un punto de vista algorítmico. Ambos autores, sin embargo, tenían en mente el mismo objetivo: proporcionar una medida de la información contenida en el mensaje que fuese independiente del sujeto que la interpreta (y por lo tanto trata de encontrar significado en ella). En los siguientes apartados discutiremos ambos enfoques.

¹ Independientemente, Solomov y Chaitin hicieron descubrimientos análogos. Sin embargo, la teoría a la que nos referiremos suele atribuirse a Kolmogorov. Aquí seguiremos esta convención.

1.2.1. Teoría estadística de la información.

Para Shannon una fuente de información es un ente abstracto que produce símbolos de un alfabeto finito. Los símbolos producidos por la fuente son una sucesión estacionaria de variables aleatorias. En otras palabras, la probabilidad de que sea producida una cierta secuencia de símbolos no cambia con el tiempo. El modelo propuesto por Shannon para una fuente es una cadena de Markov ergódica. Es decir, hay una distribución de probabilidades límite para los símbolos del alfabeto. En una **cadena de Markov** (CM) un sistema puede existir en ciertos estados (en nuestro caso un estado corresponde a la aparición de un símbolo) y existe una cierta probabilidad de pasar de un estado a otro (es decir, existe cierta probabilidad de que, después de un símbolo, aparezca otro cualquiera). Si la CM es **ergódica** las probabilidades de pasar de un estado a otro son siempre las mismas (por ejemplo, si enviamos siempre mensajes tomados de textos en español, la probabilidad de que aparezca la letra “x” permanece constante puesto que la fuente es ergódica, pero si enviamos mensajes alternadamente en inglés y en español la probabilidad de aparición de “x” cambia con el tiempo puesto que la fuente no es ergódica).

Haciendo uso de este modelo, Shannon define la **cantidad de información** (en bits) en cada símbolo s_i del alfabeto Σ de una fuente de información S como:

$$I(s_i) = -\log_2(p_i) \quad (1)$$

en donde p_i es la probabilidad de que el símbolo s_i sea producido por S y \log_2 denota el logaritmo en base 2. De esta definición se sigue que la información de una secuencia de símbolos es igual a la suma de la información de cada símbolo siempre y cuando los símbolos sean estadísticamente independientes. Por ejemplo, dada la secuencia $s_1s_2s_3$ tenemos que

$$I(s_1s_2s_3) = -\log_2(p_1p_2p_3) = -\log_2(p_1) - \log_2(p_2) - \log_2(p_3) = I(s_1) + I(s_2) + I(s_3) \quad (2)$$

En esta definición, la información contenida en un símbolo es mayor mientras menos frecuente sea el símbolo. Es decir, en la definición de Shannon, el concepto de

información está asociado con el de sorpresa. Nótese que la formulación matemática del concepto de información hace arbitraria la elección de aquello que se considere como símbolo básico. En la ecuación (2) podemos considerar, por ejemplo, al conjunto s_2s_3 como un símbolo único. El contenido informático global del mensaje (la secuencia $s_1s_2s_3$) permanece constante. Entonces, mientras más inesperado sea el evento que se observa, más información nos proporciona. Por ejemplo, si recibimos secuencias parecidas repetidamente, de acuerdo con esta definición, la información del mensaje es menor que si las secuencias no se repiten.

Como ejemplo, piense en una típica canción popular. Estas canciones normalmente consisten de un conjunto de temas que se repiten varias veces. Además, los temas de los que tratan dichas canciones normalmente son predecibles. En contraste con lo anterior, piense ahora en una obra de Bach o Beethoven. En estas obras los temas no se repiten de la misma manera y son, normalmente, difíciles de predecir. Las obras de música popular, en este sentido, nos ofrecen mucho menor información que las obras de música barroca o clásica. En otro contexto, la pintura cubista de Picasso consta de menos líneas y menos colores que la pintura barroca de, por ejemplo, Rembrandt. Picasso, puede decirse, nos da menos información en sus obras que Rembrandt. Las obras cubistas presuponen que hay más información en el observador (puesto que hay menos en las pinturas) que las obras barrocas. Por ello demandan de la búsqueda de más isomorfismos por parte del observador (por ejemplo, los rostros alterados de “Les Demoiselles d’Avignon” de Picasso solamente nos sugieren a aquéllos a los que estamos acostumbrados), en tanto que las obras pictóricas barrocas (como “La Adoración de los Pastores” de Rembrandt, en la que las imágenes son de un realismo impresionante) nos proporcionan más información de manera directa.

Habiendo definido el concepto de información de un símbolo y un mensaje procederemos a definir el concepto de **entropía de información** como el valor esperado de la

cantidad de información de los símbolos producidos por una fuente. La entropía de la fuente S queda, entonces, determinada por:

$$H(S) = \sum_{i=1}^n P_i I(s_i) = \sum_{i=1}^n p_i \log_2 \frac{1}{p_i} = - \sum_{i=1}^n p_i \log_2 p_i \quad (3)$$

La cantidad de información de un símbolo, como ya se dijo, es mayor si el símbolo es menos probable, y la entropía de la fuente es mayor mientras más uniforme sea la distribución de probabilidades de los símbolos de un alfabeto. Para cualquier fuente S con un alfabeto de n símbolos se cumple:

$$H(S) \leq \log_2 n \quad (4)$$

De hecho, en el caso de una fuente S con n símbolos equiprobables (es decir, cada uno de ellos tiene la misma probabilidad, $1/n$) se cumple la igualdad de (4). La entropía es, por lo tanto, una medida del desorden de la fuente. Cuanto más impredecible el comportamiento de S , mayor será su entropía. Por el contrario, mientras menor sea la entropía, más organizada es S . Dicho de otra manera, si $H(S)$ es pequeña significa que podemos encontrar regularidades en el fenómeno que observamos (la colección de símbolos emanados de S).

1.2.1.1. Codificación.

En la teoría de Shannon hay un elemento que está tácito, y es que los datos que constituyen el mensaje pueden ser **codificados**. Es decir, podemos establecer una relación **biunívoca** (bidireccional, sin ambigüedad de interpretación) entre el objeto de nuestro análisis (por ejemplo, las palabras de un texto, las notas de una melodía o los colores de un cuadro) y los elementos a través de los cuales representamos dichos objetos (por ejemplo, conjuntos de 0s y 1s). El almacenamiento de los datos puede hacerse de manera analógica o digital.

Por ejemplo, las grabaciones tradicionales en una cinta de carrete abierto almacenan la información de manera **analógica**: a cada aumento de volumen en el sonido grabado

corresponde un cambio en la orientación de las partículas ferromagnéticas en la cinta de grabación. El soporte sobre el que se graba la señal de audio es la cinta magnética de audio. Dicha cinta es una lámina alargada de material plástico en cuya superficie se añaden partículas ferromagnéticas capaces de retener el magnetismo inducido. La información se graba sobre el soporte cuando éste pasa delante del electroimán de la cabeza grabadora. El electroimán actúa reorientando las partículas del material ferromagnético (óxidos de hierro o de cromo) que cubren el soporte. El término “analógico” se deriva del hecho que la orientación cambia de manera análoga a las ondas de aire que constituyen el sonido original.

En contraste, cuando se almacena la información de forma **digital**, a cada sonido en cada instante del tiempo le corresponde un número (o dígito—de allí el término “digital”). Es decir, se adopta la convención de que a un cierto sonido le corresponde un cierto número. En las grabaciones digitales se dejan pulsos de información impresos que representan en el lenguaje de las computadoras (números binarios) el sonido que después va a ser leído a través del mismo sistema para poder escucharse nuevamente. Los sistemas normales de grabación digital graban en cada segundo unos 44 mil pulsos o fragmentos de información. Cada uno de los pulsos de información debe ser codificado de acuerdo con ciertas convenciones o estándares. La digitalización equivale a renunciar al valor exacto de la muestra (el pulso de información original) y a reemplazarlo por un valor numérico que lo aproxima, cuya virtud reside en que se puede representar con los n bits elegidos (por ejemplo, se puede haber escogido un valor de $n=16$). En este caso, se pierden las variaciones más pequeñas de la señal, específicamente todas las que quedan comprendidas entre dos escalones o cifras consecutivas en la representación de n bits que se haya elegido. Para que el sistema de muestreo y digitalización sea satisfactorio, es preciso que la información perdida sea irrelevante (en el caso de una grabación de sonido, inaudible). Esto se logra eligiendo la frecuencia de muestreo (es decir, no tiene que ser

exactamente 44 mil veces por segundo) de modo tal que el ancho de banda resultante incluya todas las señales audibles, y adoptando un número de bits que asegure un rango dinámico (gama de frecuencias distintas representables) igual o superior al del oído.

La pregunta que surge es ¿cómo establecemos la relación entre un dígito binario y el número o valor que se pretende representar? Para responder a esto, tocaremos el tema de la codificación.

1.2.1.2. Codificación binaria.

El sistema decimal común que usamos para representar los números es un sistema de numeración **posicional** (la posición de un dígito en un número determina su valor, puesto que cada posición tiene un peso diferente) que emplea 10 símbolos y donde la base es 10 (éste es el valor que se usa para determinar el peso de cada posición en un número), como se aprecia en el ejemplo de la Figura 12.1.

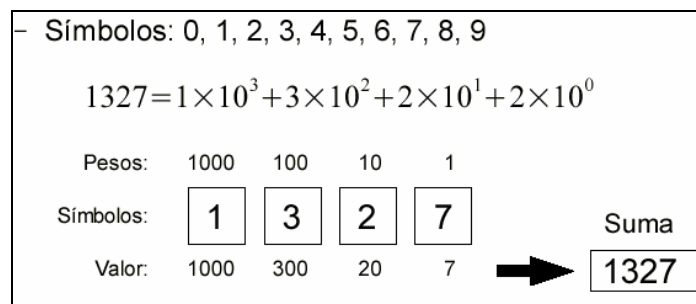


Figura 12.1. Representación decimal posicional.

Análogamente, el sistema binario (en el cual solamente existen los símbolos 0 y 1) permite representar un número cualquiera, como se ejemplifica en la Figura 12.2.

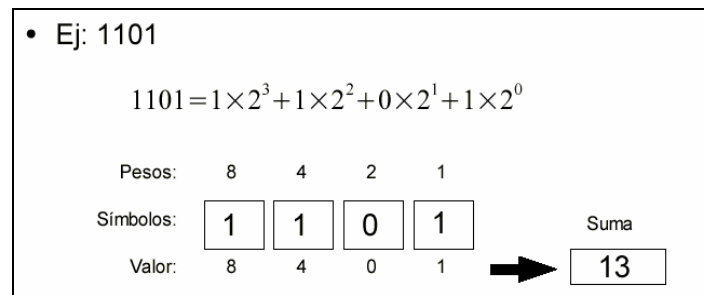


Figura 12.2. Representación binaria posicional.

El mismo concepto puede aplicarse a una base cualquiera b de la siguiente manera (donde x es el valor numérico que se desea representar/codificar, las x_i son las distintas cifras—cada una de ellas en una posición distinta—con las que se va a representar el valor x , n es la cantidad de posiciones que se tienen disponibles para representar la parte entera de x y m es la cantidad de posiciones que se tienen disponibles para representar la parte fraccionaria de x):

$$x = x_{n-1} \times b^{n-1} + \dots + x_1 \times b^1 + x_0 \times b^0 + x_{-1} \times b^{-1} + x_{-2} \times b^{-2} + \dots + x_{-m} \times b^{-m} \quad (5)$$

El mayor número entero representable con n cifras es $b^n - 1$, mientras que el mayor número real² representable es $b^n - b^{-m}$. El sistema decimal nace de hacer $b=10$ en la ecuación (5) y el sistema binario de hacer $b=2$. Ahora bien, establecer un código binario consiste en aplicar los conceptos anteriores para asignar palabras binarias a símbolos. Los símbolos pueden ser números, letras o cualquier otra entidad (información) que se quiera representar. Normalmente, cada código emplea palabras binarias de longitud fija (por ejemplo, 1 Byte=8 bits), que es lo que hace que existan las cantidades m y n en la ecuación (5). La asignación de palabras binarias en un código en particular se suele realizar para obtener alguna propiedad de interés en la información codificada. Por ello en algunos códigos no todas las palabras binarias tienen asignado un símbolo. El **código binario natural** asigna códigos binarios a números enteros positivos. La palabra

² En la computadora un número **real** está constituido por una secuencia binaria que puede interpretarse como un número con signo que consta de una parte entera y una parte fraccionaria.

asignada corresponde a la representación del número en base 2. Un ejemplo de un código binario natural de 3 bits se muestra en la Tabla 1.

Tabla 1. Asignación de códigos binarios:

Símbolo	Código
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

No todos los códigos binarios son posicionales. Algunos ejemplos alternos se muestran en la Figura 12.3. El **código exceso-3** surge de sumar 3 al número binario natural. El **código 2 de 5** posee la propiedad de usar cinco bits para representar el símbolo correspondiente, pero asignar un valor de 1 a exactamente dos de estos bits. Tiene el objetivo de detectar un número impar de errores en la transmisión de los datos. El **código de 7 segmentos** asigna un 1 a algunos de los siete segmentos activos (los que se desean prender) que forman parte de un hipotético elemento de despliegue visual (típico en carátulas de relojes digitales). Por ejemplo, el número correspondiente al símbolo 3 se codifica con 1111001 que indica que los segmentos *a*, *b*, *c*, *d* y *g* deben activarse, mientras que los segmentos *e* y *f* deben permanecer inactivos (apagados).

Cifra	natural	exceso-3	2 de 5	7 segm. abcdefg
0	0000	0011	00011	1111110
1	0001	0100	00101	0110000
2	0010	0101	00110	1101101
3	0011	0110	01001	1111001
4	0100	0111	01010	0110011
5	0101	1000	01100	1011011
6	0110	1001	10001	1011111
7	0111	1010	10010	1110000
8	1000	1011	10100	1111111
9	1001	1100	11000	1110011

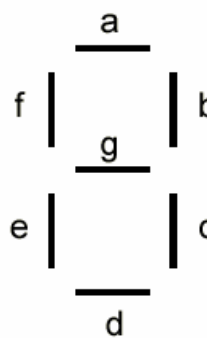


Figura 12.3. Ejemplo de códigos binarios no posicionales.

Un tipo de código binario no posicional muy interesante es el denominado **código Gray**. Este código asigna códigos binarios a números enteros positivos. La asignación se realiza de modo que la diferencia entre el código de un valor y el código del valor que le sigue sea únicamente de 1 bit (se dice que su **distancia Hamming** es 1). El código de n bits se construye de forma simétrica a partir del código de $n-1$ bits, como se ilustra en la Figura 12.4.

Número	1 bit	2 bits	3 bits	4 bits
0	0	00	000	0000
1	1	01	001	0001
2		11	011	0011
3		10	010	0010
4			110	0110
5			111	0111
6			101	0101
7			100	0100
8				1100
9				1101
10				1111
11				1110
...				...

Figura 12.4. Ilustración del código Gray.

1.2.1.3. Codificación de textos.

Para codificar textos se han adoptado ciertas convenciones que se han convertido en estándares de facto. En estos casos a cada símbolo de texto o carácter se asigna un número “equivalente” que se almacena como un valor binario. Además de símbolos gráficos se incluyen símbolos de control (tales como <nueva línea>, <nueva página>, <fin de archivo>, etc.). Existen varias asignaciones que están en uso (codificaciones). Algunos ejemplos se incluyen en la Figura 12.5, incluyendo el **código ASCII** (American Standard Code for Information Interchange), uno de los más utilizados.

ASCII (American Standard Code for Information Interchange) (ISO-646-IRV): 7 bits (128 símbolos), el más extendido, pensado para el Inglés
ISO-8859-1 (Latin 1): Extensión del ASCII, 8 bits (256 símbolos), incluye las lenguas de Europa occidental y otras.
ISO-8859-15 (Latin 9): Modificación del Latin 1 para incorporar el símbolo del EURO y otras actualizaciones.
ISO-8859-2 a ISO-8859-14: Extensiones del ASCII para diferentes lenguas: cirílico, árabe, griego, hebreo, etc.
ISO-10646 (Unicode): código de 16 bits que engloba a todas las codificaciones e idiomas conocidos.

Figura 12.5. Diferentes variantes de ASCII.

Las convenciones de ASCII para las 128 combinaciones posibles se incluyen en la Figura 12.6. En dicha figura, la primera columna corresponde al número en decimal (que se va a usar para representar un símbolo), la segunda al número hexadecimal (base 16) y la tercera al símbolo correspondiente. Cada símbolo, como se puede ver, es codificado utilizando siete bits. Quizá parezca que algunas asignaciones de código sean lógicas y otras completamente arbitrarias. Puede notarse, por ejemplo, que las letras mayúsculas (A, B, ..., Z) están codificadas por los números 65 al 90 y que los dígitos numéricos (los caracteres 0, 1, ..., 9) están codificados por los números 48 al 57. Las preguntas que ahora planteamos son las siguientes. ¿Es ésta la mejor forma de codificar las letras y símbolos de un texto? ¿Será posible aprovechar el conocimiento de la entropía de una fuente que corresponda, digamos, al idioma español, al elegir los códigos correspondientes? Las respuestas a estas dos preguntas son “no” y “sí”, respectivamente.

0 00 NUL	32 20 SPC	64 40 @	96 60 `
1 01 SOH	33 21 !	65 41 A	97 61 a
2 02 STX	34 22 "	66 42 B	98 62 b
3 03 ETX	35 23 #	67 43 C	99 63 c
4 04 EOT	36 24 \$	68 44 D	100 64 d
5 05 ENQ	37 25 %	69 45 E	101 65 e
6 06 ACK	38 26 &	70 46 F	102 66 f
7 07 BEL	39 27 '	71 47 G	103 67 g
8 08 BS	40 28 (72 48 H	104 68 h
9 09 HT	41 29)	73 49 I	105 69 i
10 0A LF	42 2A *	74 4A J	106 6A j
11 0B VT	43 2B +	75 4B K	107 6B k
12 0C FF	44 2C ,	76 4C L	108 6C l
13 0D CR	45 2D -	77 4D M	109 6D m
14 0E SO	46 2E .	78 4E N	110 6E n
15 0F SI	47 2F /	79 4F O	111 6F o
16 10 DLE	48 30 0	80 50 P	112 70 p
17 11 DC1	49 31 1	81 51 Q	113 71 q
18 12 DC2	50 32 2	82 52 R	114 72 r
19 13 DC3	51 33 3	83 53 S	115 73 s
20 14 DC4	52 34 4	84 54 T	116 74 t
21 15 NAK	53 35 5	85 55 U	117 75 u
22 16 SYN	54 36 6	86 56 V	118 76 v
23 17 ETB	55 37 7	87 57 W	119 77 w
24 18 CAN	56 38 8	88 58 X	120 78 x
25 19 EM	57 39 9	89 59 Y	121 79 y
26 1A SUB	58 3A :	90 5A Z	122 7A z
27 1B ESC	59 3B ;	91 5B [123 7B {
28 1C FS	60 3C <	92 5C \	124 7C
29 1D GS	61 3D =	93 5D]	125 7D }
30 1E RS	62 3E >	94 5E ^	126 7E ~
31 1F US	63 3F ?	95 5F _	127 7F DEL

Figura 12.6. Codificaciones ASCII.

Una de las aplicaciones más directas de los resultados derivados de la teoría estadística de la información es que es posible determinar, de manera exacta, la mejor forma de elegir los códigos dependiendo de las características de S . Para ello es indispensable abandonar la idea de asignar códigos de longitud igual a cada símbolo. Se deben asignar códigos de menor longitud a los símbolos más probables y de mayor longitud a los menos probables. Esta observación es intuitiva y la encontramos en el **código Morse**. Morse, actuando de manera pragmática antes de la teoría de Shannon, determinó las longitudes de su código contando el número de letras en las cajas de tipos usados por los impresores para componer las páginas de textos en inglés. Pero él no sabía, como lo sabemos nosotros ahora, que es posible obtener una

asignación de códigos binarios tal que la longitud promedio de un símbolo puede aproximarse a la entropía de la fuente. Más aún, Shannon demostró que la longitud promedio de un código nunca puede violar la desigualdad (4).

1.2.1.4. Codificación de imágenes.

Los resultados de Shannon (y de la teoría de la información, en general) no están limitados a la representación de textos. En la computadora podemos codificar también, como ejemplo, imágenes. Las computadoras componen imágenes mediante el dibujo de puntos de distinto color denominados **píxeles**. El color de cada punto se codifica con un número binario usando una cantidad determinada de bits. La cantidad de bits empleada se llama **profundidad de color**. Por ejemplo, se podrían usar 8 bits, 16 bits, 24 bits (un código llamado **color verdadero**), etc. Un ejemplo de un píxel se muestra en la Figura 12.7.

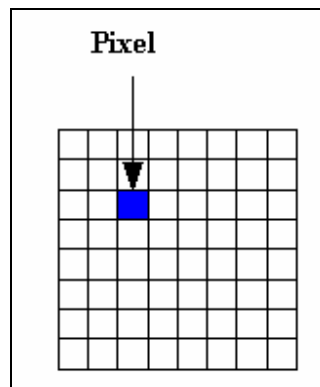


Figura 12.7. Ejemplo de un píxel.

Un detalle de una imagen de color se muestra en la Figura 12.8. Para representar imágenes como ésta, se debe hacer una codificación del color. Un determinado color se forma combinando tres colores primarios en distintas intensidades, el rojo (R), el verde (G) y el azul (B). Si se utilizan 24 bits en total para representar el color de un píxel, se asignan 8 de estos bits para cada color primario, por lo cual existe un total de 256 valores diferentes de intensidad (0 a

255) para cada color primario y más de 16 millones de colores distintos que se pueden producir combinando dichos valores de los colores primarios.



Figura 12.8. Imagen compuesta por pixeles (del lado derecho, detalle de un fragmento de la imagen).

Algunos ejemplos de la asignación de colores en base a combinaciones de valores para los colores primarios se muestran en la Figura 12.9.


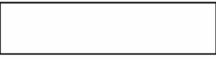




R G B	
	0 0 0 negro
	255 255 255 blanco
	255 0 0 rojo intenso
	255 255 0 amarillo
	192 192 255 azul claro
	255 128 0 naranja

Figura 12.9. Algunos ejemplos de asignación de colores.

Como ejemplo, si tenemos una imagen de 20x20 pixeles requerimos de 400 valores binarios diferentes para determinar la posición de cada uno de ellos. Si además usamos 24 bits para definir el color de cada uno de los pixeles, resulta que una imagen con estas características

requerirá de $\lceil \log_2(400) \rceil + 24 \times 400 = (9 + 24) \times 400 = 13,200$ bits para ser representada. Una imagen con una resolución típica (de 800x600 pixeles) requerirá de $\lceil \log_2(800 \times 600) \rceil + 24 \times 480,000 = (19 + 24) \times 480,000 = 20,640,000$ bits para ser representada. Lo importante es que, al igual que en el caso de los textos, un conjunto de colores se convierte en una colección de bits, que pueden aprovecharse las características de la fuente que genera las imágenes y que, por ello, se aplican los mismos resultados ya mencionados anteriormente.

1.2.1.5. Codificación de sonidos.

El problema de la codificación de sonidos puede atacarse de la siguiente manera. El sonido se representa por una curva de amplitud (que puede representar la presión de las ondas de aire sobre la cóclea, una señal eléctrica, etc.) frente al tiempo. Esto se ilustra en la Figura 12.10. La señal debe codificarse digitalmente.

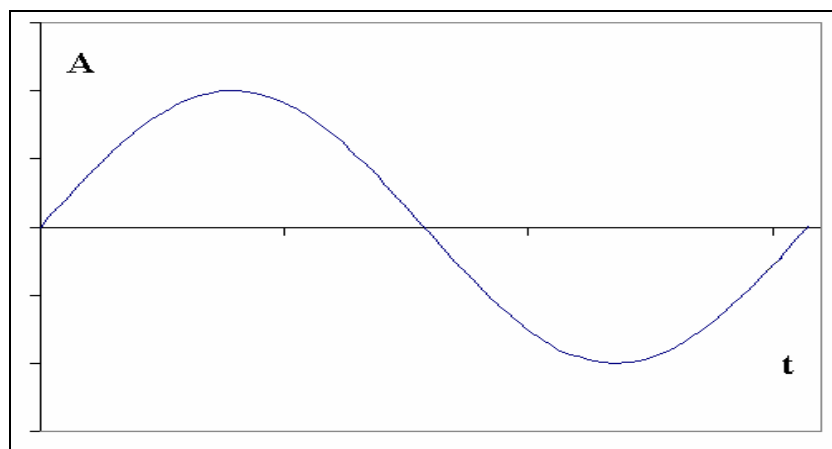


Figura 12.10. Amplitud vs. tiempo.

La codificación digital implica dos pasos, muestreo y cuantización. El **muestreo** implica que se mide el valor de la señal a intervalos regulares, como se ilustra en la Figura 12.11. Las frecuencias típicas de muestreo son 11025, 22050 o 44100 Hz (veces por segundo). La **cuantización** implica que la amplitud se representa usando una cantidad determinada de bits (por ejemplo, 8,16 o 24). La calidad de la codificación será mayor cuanto más grandes sean la

frecuencia de muestreo y la cantidad de bits por muestra. Por ejemplo, la calidad de un CD se obtiene muestreando a una frecuencia de 44100 Hz, usando 16 bits por muestra. Con estas características, una melodía de 2:30 minutos (150 segundos) de duración requerirá de $44100 \times 16 \times 150 = 105,840,000$ bits para ser codificada. ¡Esto equivale a 13,230,000 Bytes (13.23 MBytes), que es aproximadamente la cantidad de caracteres que contiene un texto de 6600 páginas!

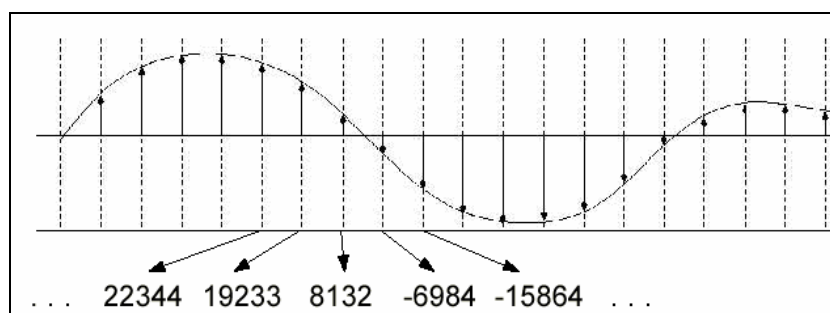


Figura 12.11. Muestreo de una señal de audio.

1.2.1.6. Consideraciones finales.

De lo anteriormente expuesto es relativamente fácil inferir que:

- a) Todo aquello que es susceptible de ser cuantificado también es susceptible de ser codificado.
- b) Aquello que es codificable puede codificarse en algún código binario, lo que implica que:
 - c) Todo tipo de datos puede ser expresado usando números binarios, y por lo tanto:
 - d) Las conclusiones derivadas de la teoría de la información pueden ser aplicables a cualquier tipo de datos.
 - e) La información contenida en un conjunto de datos puede ser cuantificada de manera precisa.

f) Existe una forma óptima de codificar los datos.

Estas conclusiones explican, en buena medida, el interés que hay en estudiar esta teoría, particularmente en el contexto del manejo eficiente de datos. En la época actual, en la que las comunicaciones a través de Internet se han incrementado de forma espectacular, se hace aún más importante abordar el problema de encontrar la forma más económica y adecuada de transmitir y almacenar los datos sin perder información (en inglés, **lossless transmission**) o, en caso de que cierta cantidad de pérdida sea aceptable, perdiendo parte de la información (en inglés, **lossy transmission**). Ambos casos de representación, sin pérdida (lossless) y con pérdida (lossy) se discutirán más adelante.

1.2.2. Teoría algorítmica de la información.

Ya que la teoría estadística de la información (TEI) nos dice que, sin importar el origen de los datos, podemos medir la cantidad de información contenida en ellos así como su entropía y que, como consecuencia de ello, podemos encontrar la codificación óptima, nos encontramos ante una teoría que es a la vez elegante y precisa. Sin embargo, hay tres puntos relacionados con la TEI que vale la pena discutir.

El primero de ellos tiene que ver con el hecho de que en la TEI se supone que se conocen las probabilidades p_i de los símbolos s_i de los mensajes. Normalmente este no es el caso. Las probabilidades se aproximan por las proporciones de cada símbolo en un mensaje dado. Por ejemplo, la probabilidad de aparición de la letra “a” en un texto escrito en español puede aproximarse contando la cantidad de veces que dicha letra aparece en el mensaje y dividiendo esto entre el número total de caracteres del mensaje. Como ejemplo, si se toma como mensaje el texto del Capítulo 13 de este libro (el “mensaje” 1), resulta que la letra “a” aparece 5306 veces en un total de 113153 caracteres y que, por lo tanto, $p(\text{“a”})=0.04689$. Por otro lado, un análisis

similar para el texto correspondiente al presente capítulo (el “mensaje” 2) nos dice que “a” aparece 2608 veces en un total de 207360 caracteres, y por lo tanto $p("a")=0.01258$. Como se ve, las probabilidades son distintas y esto es indicativo de las variaciones tanto del carácter en cuestión dentro del texto como del contexto en el que estos caracteres se encuentran. En la Figura 12.12 se ilustra lo anterior. En dicha figura se observan cuatro recuadros. Los dos primeros corresponden al mensaje 1. En el recuadro izquierdo, se ve el carácter ASCII, su representación decimal y el número de veces que aparece en el archivo. En el recuadro derecho se ven los mismos caracteres, su representación, la probabilidad estimada y la cantidad de información correspondiente (medida en bits). Los últimos dos recuadros de la figura, de manera análoga, se refieren al mensaje 2.

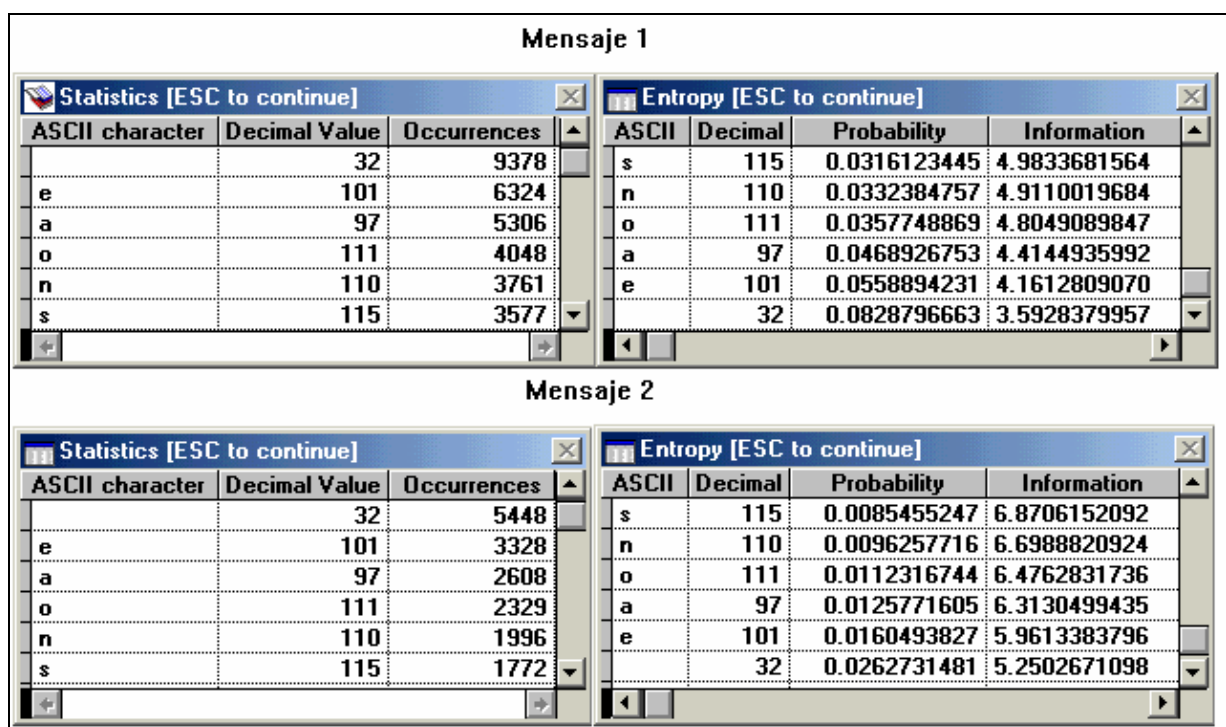


Figura 12.12. Estadísticas para dos mensajes de texto.

El segundo punto tiene que ver con la elección de los símbolos básicos de nuestro alfabeto. Como ya se vio, en esta teoría es posible suponer que los átomos del alfabeto son bits, o parejas de bits, o grupos de tres bits, etc. La teoría nos permite elegir la longitud de las secuencias

de bits para usarlas como símbolos siempre y cuando los bits de los símbolos elegidos sean estadísticamente independientes. Normalmente este no es el caso y, en la aplicación de la TEI, hay que hacer aproximaciones, como se discutirá más adelante.

El tercer punto es más sutil, pero igualmente interesante. Resulta que de acuerdo con la TEI, el transmisor y el receptor deben saber a priori (de antemano) cuál es el universo (el conjunto total) de posibles mensajes a transmitir. Por ejemplo, cuando nosotros enviamos texto en ASCII, tácitamente suponemos que cada carácter consta de siete bits y que, además, conocemos el significado de cada combinación binaria (de esos siete bits). Pero podemos componer mensajes de mucho mayor complejidad (de los cuales no sabemos nada) a partir de los átomos de información (caracteres) acordados. Para ilustrar la limitación referida pensemos en dos interlocutores a los que llamaremos Tex y Rex. Tex y Rex son estudiosos de las religiones del mundo y deciden que solamente les interesa transmitirse entre sí la Biblia, el Corán, los Vedas (cuyos contenidos ambos conocen) y el saludo “¡Buenos días!” como un acto de cortesía. Ambos saben codificar en binario y saben, por tanto, que las cuatro combinaciones posibles que existen de dos bits son suficientes para indicar cuál de las cuatro opciones (mensajes) se ha elegido. Más aún, asignan el código “00” a la Biblia, el “01” al Corán, el “10” a los Vedas y el “11” al saludo. De manera que la secuencia “110010” que Tex transmite y Rex recibe significa “¡Buenos Días! Estudiemos la Biblia y después los Vedas”. Evidentemente el significado de la Biblia o los Vedas, en este caso, es mucho mayor y más complejo que el saludo “¡Buenos días!”. Pero para la TEI “¡Buenos días!” o el texto íntegro del Corán llevan la misma información, exactamente dos bits. En este ejemplo extremo solamente hay cuatro símbolos, los cuales representan el texto de la Biblia, el texto del Corán, el texto de los Vedas y el texto del saludo. Normalmente no esperaríamos que nadie mantuviera una conversación usando solamente cuatro símbolos ni que los símbolos de nuestro alfabeto estuvieran constituidos por textos completos (y

menos de la complejidad de los textos del ejemplo), pero el ejemplo es válido independientemente de que sea poco realista.

1.2.2.1. Información según Kolmogorov.

Para Kolmogorov, como alternativa al enfoque de Shannon, existe una manera de medir la información contenida en un mensaje que es independiente del observador. Definió información como la longitud, medida en bits, del programa más corto capaz de reproducir el mensaje. Pensemos en el mensaje binario $\underbrace{111\dots\dots\dots 11}_{100,000,000 \text{ veces}}$. Este mensaje puede ser

reproducido, fácilmente, con el programa de la Figura 12.13 (escrito en pseudocódigo):

```
for i=1 to 100000000
    print "1"
endfor
```

Figura 12.13. Programa que produce cien millones consecutivos de 1's.

Si usamos un carácter ASCII para representar cada una de las letras, espacios y signos de control (como el **regreso de carro**³) del programa, veremos que son necesarios 37x7 o 259 bits. Esa, de acuerdo con Kolmogorov es la **información algorítmica** (IA, también conocida como **complejidad de Kolmogorov** o CK) contenida en una secuencia de cien millones de 1s. Por el contrario, pensemos ahora en el texto íntegro de la Biblia. El programa capaz de reproducirlo (sin entrar en detalles) es claramente mucho mayor que el que se muestra arriba y, por lo tanto, contiene mucho más información, aún suponiendo que ambos textos tuvieran los mismos 100 millones de bits. Lo que se puede observar en esta teoría, es que los patrones (los

grupos de datos con cierta estructura) contenidos en el mensaje le restan información al mismo. Por ejemplo, el mensaje “ABCDABCDABCDABCD” puede programarse más brevemente que el mensaje “0123456789ABCDEF”, aunque ambos tengan el mismo número de caracteres, puesto que en el primer mensaje es fácil identificar el patrón “ABCD” repetido 4 veces. Por ello el primer mensaje contiene menos información que el segundo. Esta observación nos conduce a una conclusión por demás interesante. Si un mensaje contiene la máxima cantidad posible de información, la longitud del programa que sea capaz de reproducirlo será igual a la longitud del mensaje mismo. Esto solamente ocurre si no existen patrones de repetición de ningún tipo en el mensaje, lo cual equivale a decir que el mensaje es totalmente irregular o **aleatorio**. De hecho, Chaitin (Chaitin, 2004) ha definido una secuencia como aleatoria si y solo si la información algorítmica contenida en ella es igual a la longitud de la secuencia misma.

Para que la definición de información algorítmica (IA) tenga un sentido general, es indispensable establecer el lenguaje de programación que se considera cuando se desea encontrar el programa más corto. Kolmogorov apeló a conceptos de teoría de la computación, estableciendo que la máquina en la que se va a programar el algoritmo es la denominada máquina de Turing (véase el Capítulo). Esta “máquina”⁴ solamente puede programarse de una manera y con un lenguaje básico y satisface todos los requerimientos de computabilidad estipulados por la definición de Kolmogorov. Pero puede demostrarse (Li and Vitányi, 1997) que la IA es independiente del lenguaje en el que se programe excepto por una constante aditiva. Es decir, si encontramos el programa más corto correspondiente al mensaje M en el lenguaje C_1 y también lo hacemos en el lenguaje C_2 , la información algorítmica para C_1 será $I+K_1$ y aquella para C_2 será

³ En ASCII el regreso de carro indica que debemos movernos al inicio de la siguiente línea y corresponde al código 0001101, es decir, al número 13.

$I+K_2$, de manera que no es muy importante en qué lenguaje decidamos encontrar la descripción más corta, siempre y cuando siempre usemos el mismo lenguaje para todos nuestros cálculos. La teoría algorítmica de la información (TAI) nos ofrece un resultado negativo fundamental: en general es imposible calcular la IA para un mensaje arbitrario en un tiempo acotado. Es decir, aunque en principio es posible calcular la IA para cualquier mensaje, el tiempo que requiere realizar dicho cálculo no puede determinarse. En otras palabras, requeriríamos de una computadora infinitamente rápida para obtener la IA de un mensaje arbitrario en un tiempo corto. Esto no significa que nunca pueda calcularse la IA, sino que hay casos en los que la determinación de la cantidad de información es extremadamente compleja.

1.2.2.2. Expresiones polinomiales.

Para efectos de los ejemplos que siguen, adoptaremos una convención que nos permite calcular, aunque sea aproximadamente, la IA de un mensaje. En la Tabla 2 aparecen diez valores de x y, para cada uno de ellos, el correspondiente valor de la función $y=f(x)$. ¿Qué tanta información algorítmica se encuentra contenida en esta tabla? Si fuéramos capaces de expresar esta tabla de la manera más compacta posible, estaríamos encontrando la IA. A primera vista no se discierne un patrón de repetición obvio.

⁴ En realidad la “máquina” de Turing es una abstracción matemática ideada por el matemático Alan Turing que no puede ser construida, pero nos ofrece un marco teórico que nos permite precisar lo que entendemos como “el programa más corto”.

Tabla 2. Valores de una función $y=f(x)$:

x	$f(x)$
-1.3000	-5.2500
4.0000	8.0000
0.5000	-0.7500
2.7183	4.7958
3.1416	5.8540
0.6180	-0.4549
-5.250	-15.250
-0.750	-3.8750
5.8540	12.6350
-0.4549	-3.1373

Aplicando técnicas de análisis numérico (Shampine, Allen, and Pruess, 1997) es fácil encontrar que la expresión de la ecuación (6) nos permite conocer cada uno de los valores de y para una x cualquiera:

$$y = 2.5x - 2 \quad (6)$$

De esta manera, los números 2.5 y -2 son suficientes para expresar el contenido total de la Tabla 2 (que, en este caso, es nuestro mensaje) si suponemos que vamos a expresar los mensajes como una suma de monomios. La suma “ $2.5x^1 + (-2)x^0$ ” constituye un **polinomio**. Cada monomio (cada término del polinomio) en este caso consiste de un coeficiente (cantidad numérica) multiplicando a una potencia entera de la variable x . En un lenguaje de programación típico cada número real ocupa 32 bits. La Tabla 2 consta de 10 filas y 2 renglones. Para representar a cada uno de los elementos de la Tabla 2, entonces, requerimos de $10 \times 2 \times 32 = 640$ bits. Pero el mensaje en cuestión podría ser simplemente expresado con los 64 bits correspondientes a los dos coeficientes de la expresión polinomial equivalente. Es decir, la información algorítmica de la Tabla 2 es de 64 bits (no de 640 bits, como podría pensarse de no haber encontrado los patrones que permiten representar la tabla como una expresión polinomial). Regresando al ejemplo de los cien millones de 1s consecutivos, se puede ver que el polinomio “ $y=I$ ” nos permite obtener cada uno de los valores del mensaje, ya que se cumple que $y_i=I$ para

$x_1, x_2, \dots, x_{100000000}$. Adoptando la convención polinomial, la CK de una secuencia consecutiva de cien millones de 1s es de 32 bits.

1.2.3. Un ejemplo de cálculo de información.

Para mejor ilustrar los conceptos anteriores, haremos ahora el ejercicio de calcular la información contenida en dos figuras geométricas muy simples (el mensaje) que se muestran en las Figuras 12.14 y 12.15, respectivamente. Las figuras consisten de un triángulo y un cuadrado. Cada una de ellas se ha ubicado dentro de una matriz de 20x20 píxeles. Es decir, hay un total de 400 píxeles por figura. Como no estamos interesados en manejar colores, solamente necesitamos de dos símbolos. Se va a utilizar un 0 para representar un “ ” (espacio en blanco) y un 1 para representar un “+” (que, como se puede ver en las figuras, es el único otro carácter que se necesita representar). Para facilidad de codificación, en las figuras se incluyen dos “reglas” (ejes) cuyos números nos permiten identificar las coordenadas de cada píxel. Estos números no forman parte de la imagen que deseamos analizar.

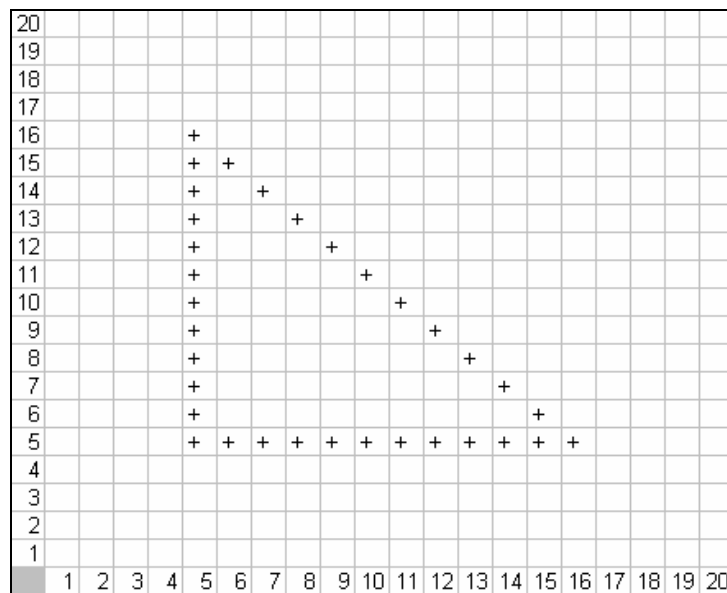


Figura 12.14. Un triángulo.

20																				
19																				
18																				
17																				
16				+	+	+	+	+	+	+	+	+	+	+	+					
15				+														+		
14				+														+		
13				+														+		
12				+														+		
11				+														+		
10				+														+		
9				+														+		
8				+														+		
7				+														+		
6				+														+		
5				+	+	+	+	+	+	+	+	+	+	+	+	+				
4																				
3																				
2																				
1																				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Figura 12.15. Un cuadrado.

Lo que haremos es representar cada figura usando una tabla de 400 filas. Cada fila consta de tres elementos, la coordenada horizontal (x), la coordenada vertical (y) y el valor del píxel correspondiente (z , cuyo valor puede ser 0 o 1). El origen de las coordenadas está en la esquina inferior izquierda de la matriz mostrada en cada figura. Así, por ejemplo, el “+” en el vértice superior del triángulo de la Figura 12.14 se representa como (5, 16, 1), en tanto que el píxel inmediatamente a su derecha se describe con (6, 16, 0). De manera análoga, el “+” en la esquina inferior derecha del cuadrado de la Figura 12.15 se representa como (16, 5, 1), en tanto que el píxel inmediatamente a su izquierda es (15, 5, 1). Estas convenciones dan origen a una matriz de 3 x 400 elementos para cada una de las figuras. Una pequeña parte de la tabla del triángulo se incluye en la Tabla 3.

Tabla 3. Segmento de la tabla de coordenadas para el triángulo:

X	Y	Z
14	6	0
15	6	1
16	6	0
17	6	0
18	6	0

19	6	0
20	6	0
1	5	0
2	5	0
3	5	0
4	5	0
5	5	1
6	5	1
7	5	1

1.2.3.1. La información de acuerdo con la teoría estadística.

Para calcular la información que contienen el triángulo y el cuadrado de acuerdo con la teoría de Shannon, lo que debemos hacer es:

- Estimar la información de cada uno de los símbolos (en este caso 0 y 1).
- Obtener la información inherente a cada símbolo.
- Sumar la información de cada símbolo.
- Obtener la entropía para el mensaje.

Para esto podemos escribir un programa que efectúe estos cálculos para nosotros. En la Figura 12.16 mostramos los resultados de ejecutar dicho programa. Se puede notar que la información contenida en el triángulo es ligeramente menor que aquella contenida en el cuadrado. Esto se debe a que la entropía en el cuadrado es mayor, lo cual es intuitivamente satisfactorio, puesto que el triángulo solamente tiene tres vértices, en tanto que el cuadrado tiene cuatro. Nos “dice” más un cuadrado que un triángulo.

Información Estadística		
	Triángulo	Cuadrado
1's	33	44
0's	367	356
P0	0.9175	0.8900
P1	0.0825	0.1100
I0	0.4526	0.4830
I1	2.8614	2.5737
Info	260.5384	285.2113
H	0.6513	0.7130
Fin		

Figura 12.16. La información de acuerdo con la teoría de Shannon.

1.2.3.2. La información de acuerdo con la teoría algorítmica.

El ejercicio análogo para la complejidad de Kolmogorov es bastante más complejo. Pretender expresar, con un polinomio, las relaciones entre el triángulo y el cuadrado implica que se tiene que obtener una expresión matemática de la forma:

$$z(x, y) = \sum_{i=0}^{g_1} \sum_{j=0}^{g_2} c_{ij} x^i y^j \quad (7)$$

donde g_1 y g_2 denotan los grados máximos de las variables en el polinomio. Uno de los resultados de la teoría de la aproximación de Fourier (Scheid, 1968) es que cualquier función puede aproximarse con mayor precisión mientras más alto es el grado del polinomio de aproximación. En este caso, la elección del grado máximo de nuestro polinomio de aproximación determinará qué tan precisamente seremos capaces de encontrar los coeficientes que determinen la información contenida en el mensaje. Este hecho parece ser poco satisfactorio porque, entonces, dependiendo de cómo elijamos g_1 y g_2 , obtendremos distintas medidas de la información contenida en el mensaje. Para efectos de este ejemplo, elegimos $g_1 = g_2 = 4$. Vemos en la ecuación (7) que es necesario determinar los valores de 25 coeficientes (los índices corren desde 0 hasta 4

para cada una de las dos variables, por lo que los coeficientes van del c_{00} hasta el c_{44}). Los valores obtenidos para el triángulo, en este caso, se ilustran en la Tabla 4.

Tabla 4. Coeficientes para el triángulo con $g_1=g_2=4$:

C00	-1.2461	C20	-0.0239	C40	0.0000
C01	0.6215	C21	0.0099	C41	0.0000
C02	-0.0928	C22	-0.0018	C42	0.0000
C03	0.0063	C23	0.0001	C43	0.0000
C04	-0.0002	C24	0.0000	C44	0.0000
C10	0.3326	C30	0.0008		
C11	-0.1244	C31	-0.0003		
C12	0.0206	C32	0.0001		
C13	-0.0015	C33	0.0000		
C14	0.0000	C34	0.0000		

Se puede ver que, de los 25 coeficientes, el proceso de aproximación arroja 9 valores iguales a cero, por lo que, en realidad, solamente hay 16 coeficientes. Por lo tanto la información algorítmica corresponde a $16 \times 32 = 512$ bits. En este caso el error RMS⁵ de aproximación es de 0.4474. Si escogemos ahora, $g_1=g_2=5$, el número de coeficientes crece a 36, de los cuales 17 no son ceros, lo cual resulta en $IA=17 \times 32=554$ bits y $e_{RMS}=0.4392$. Por último, si $g_1=g_2=6$, el número de coeficientes diferentes de cero crece a 24, con $IA=24 \times 32=768$ bits y $e_{RMS}=0.3810$. Un análisis análogo puede hacerse para la figura del cuadrado. En la Tabla 5 se presentan los errores RMS para polinomios de grados (4,4), (5,5) y (6,6).

Tabla 5. Error RMS para distintos grados polinomiales:

Grados	4,4	5,5	6,6
e_{RMS} Triángulo	0.4774	0.4392	0.3810
e_{RMS} Cuadrado	0.4183	0.4183	0.4024

De manera análoga, en la Tabla 6 se presenta la cantidad de coeficientes efectivos (coeficientes diferentes de cero).

Tabla 6. Cantidad de coeficientes efectivos para distintos grados polinomiales:

⁵ El error RMS (del inglés root mean square, la raíz del cuadrado de los promedios) se obtiene de la

siguiente fórmula: $e_{RMS} = \sqrt{\frac{1}{n} \sum_{i=1}^n [z_i - f_i(x,y)]^2}$, en donde z_i es el valor de la variable dependiente z y $f(x,y)$ es el valor aproximado por el polinomio.

	Coeficientes efectivos		
Triángulo	16	17	24
Cuadrado	18	20	27

Por último, en la Tabla 7 se presentan las informaciones algorítmicas para cada combinación de coeficientes.

Tabla 7. Información algorítmica:

		Información	
Grados	4,4	5,5	6,6
Triángulo	512	554	768
Cuadrado	576	640	864

¿Significa este resultado que la información no es la misma aunque las figuras sean iguales? Por supuesto que no. Lo que observamos es que mientras la descripción del mensaje se hace más precisa, vamos siendo capaces de extraer más información del mismo. En la Tabla 8 se presentan los coeficientes efectivos para el proceso de aproximación de mayor precisión en el caso del cuadrado ($g_1=g_2=6$).

Tabla 8. Coeficientes efectivos para el cuadrado:

C00	2.8961	C13	-0.0138	C30	-0.0256
C01	-2.3544	C14	0.0026	C31	-0.0007
C02	0.5360	C15	-0.0001	C32	0.0070
C03	-0.0592	C20	0.3520	C33	-0.0019
C04	0.0036	C21	-0.0989	C34	0.0002
C05	-0.0001	C22	-0.0342	C40	0.0010
C10	-2.0467	C23	0.0129	C41	0.0003
C11	1.1306	C24	-0.0014	C42	-0.0004
C12	-0.1014	C25	0.0001	C43	0.0001

1.2.3.3. Conclusiones.

Se han presentado dos maneras de definir y medir la información contenida en un mensaje. Al parecer las dos teorías (la estadística, de Shannon, y la algorítmica, de Kolmogorov) nos llevan a conclusiones distintas. Cuando medimos la información contenida en los mismos mensajes (dos figuras geométricas simples) con la TEI y luego con la TAI obtenemos resultados

diferentes. ¿A qué obedece esta aparente incongruencia de las dos teorías? ¿Es que una es correcta y la otra equivocada? Si es así, ¿cuál es la correcta?

Resulta que, como es de esperarse en dos teorías matemáticamente fundamentadas, ambas nos llevan a resultados coherentes. Por ejemplo, en ambos casos el cuadrado contiene más información que el triángulo, y en ambos casos los datos más redundantes (aquéllos que contienen la mayor cantidad de patrones) tienen menos información. En el caso de Shannon, esto se traduce en menor entropía, mientras que en el caso de Kolmogorov se traduce en menos coeficientes efectivos. Cuando los símbolos son equiprobables, la entropía se maximiza en la TEI, en tanto que en la TAI se hace evidente la aleatoriedad de los mensajes al no poderse comprimir. En el caso de la teoría estadística, puesto que conocemos el universo de los mensajes de antemano, al obtener las aproximaciones a las probabilidades de los símbolos lo que estamos haciendo es “repartir” en cada uno de los símbolos el conocimiento que tenemos acerca de ellos. Se logra una mayor generalidad, pero una menor precisión. Por contraste, en la teoría algorítmica, mientras mayor precisión se tenga en la descripción del mensaje, más información se “exprime” (extrae) de él.

Puesto que ambas teorías arrojan resultados coherentes entre ellas y, además las teorías son complementarias, ¿cuál se debe usar? y ¿qué ventajas nos reporta cada una de ellas? En la siguiente sección nos referiremos al muy interesante e importante problema de la **compresión de datos**, la representación de datos de la manera más eficiente. Veremos cómo, de manera natural, la teoría estadística nos proporciona herramientas para hacer compresión sin pérdida (lossless) mientras que la teoría algorítmica nos las proporciona para lograr la compresión con pérdida (lossy). Esto no significa que una u otra sean mejores para alguna de estas tareas, sino que hemos elegido un ejemplo de aplicación para cada una de ellas.

1.2.4. Compresión de datos.

Una de las aplicaciones más interesante de la teoría de la información es la referente a la manera más eficaz de representar un conjunto de datos arbitrario. En esta sección se presentan dos ejemplos que ilustran estos conceptos. El primero de los ejemplos está orientado a la representación sin pérdida, y para ello se aplican los denominados **códigos de Huffman** (Huffman, 1952). El segundo ejemplo se refiere a la representación más compacta de información con pérdida. Para ello se aplican los conceptos de representación polinomial tocados en la sección de teoría algorítmica de la información de este capítulo.

1.2.4.1. Compresión sin pérdida.

Empezamos esta sección considerando el mensaje que consiste de la siguiente cadena de símbolos:

“MSMQETQVQHFRKDRQARRAERAQAKDNRFQAHQQTIQMRDMEKQVKKQ
YKDHHQMKESETHVDNIKKDDKHAIEEDQMDKDNRTSYKSHIIRHLQMHTQQHQY
MQHQDQAKKEKTHGAIKGALKA”

Este mensaje se denominará “P1”. Si se cuentan los símbolos contenidos en P1, se encuentra que hay 128 de ellos, distribuidos como se muestra en la Tabla 9. Los 17 símbolos de la tabla han sido ordenados de menor a mayor de acuerdo con la cantidad de veces que aparecen en el mensaje. El símbolo menos frecuente es la letra “F” (empatando con “G” y “L”), que aparece dos veces, mientras que el más frecuente es la letra “Q”, que aparece 21 veces. Ya que desconocemos las distribuciones de probabilidad de los símbolos que aparecen en P1, las aproximaremos, como es costumbre, con las proporciones correspondientes que se muestran en la columna de la derecha de la Tabla 9.

Tabla 9. Estadísticas del mensaje P1:

Símbolo	Veces	Probabilidad
F	2	0.015625
G	2	0.015625
L	2	0.015625
N	3	0.023438
V	3	0.023438
Y	3	0.023438
S	4	0.031250
I	6	0.046875
T	6	0.046875
E	8	0.062500
M	8	0.062500
R	9	0.070313
A	10	0.078125
D	12	0.093750
H	12	0.093750
K	17	0.132813
Q	21	0.164063

Si consideramos de esta manera las probabilidades, podemos calcular la cantidad de información representada por cada símbolo y el promedio de bits de información contenido en cada símbolo, así como la entropía del mensaje. En la Tabla 10 se pueden observar estos resultados. Para este mensaje fácilmente se puede calcular la entropía que, de acuerdo con la ecuación (3), tiene un valor de $H(P_1)=3.74377$. Ahora bien, de acuerdo con la codificación ASCII de la Figura 12.6, a la letra “F” le corresponde el código hexadecimal 46 (1000110), a la “N” el hexadecimal 4E, etc. Como se puede ver en la Figura 12.5, a cada uno de los códigos ASCII de P1 le corresponden 7 bits, por lo que la longitud total del mensaje binario codificado en ASCII es de $7 \times 128 = 896$ bits. La idea de los códigos de Huffman es elegir el código más adecuado considerando la probabilidad de aparición del símbolo de manera que la longitud promedio \bar{L} de los símbolos se minimice. Para cualquier código de longitud fija k , $\bar{L} = k$. ¿Cómo elegimos los códigos para lograr lo anterior?

Tabla 10. Información de los símbolos en P1:

Símbolo	Veces	Probabilidad	Información	Bits Promedio/Símbolo
F	2	0.015625	6.000000	0.09375
G	2	0.015625	6.000000	0.09375
L	2	0.015625	6.000000	0.09375
N	3	0.023438	5.415037	0.12691
V	3	0.023438	5.415037	0.12691
Y	3	0.023438	5.415037	0.12691
S	4	0.031250	5.000000	0.15625
I	6	0.046875	4.415037	0.20695
T	6	0.046875	4.415037	0.20695
E	8	0.062500	4.000000	0.25000
M	8	0.062500	4.000000	0.25000
R	9	0.070313	3.830075	0.26930
A	10	0.078125	3.678072	0.28735
D	12	0.093750	3.415037	0.32016
H	12	0.093750	3.415037	0.32016
K	17	0.132813	2.912537	0.38682
Q	21	0.164063	2.607683	0.42782

1.2.4.2. El método de Huffman.

La idea encontrada por Huffman es la siguiente. Se ejemplifica el concepto usando un alfabeto hipotético de cuatro símbolos (que se denotan con s_1 , s_2 , s_3 y s_4). Sus probabilidades son las siguientes: $p(s_1)=0.5$, $p(s_2)=0.25$, $p(s_3)=0.125$ y $p(s_4)=0.125$. Las informaciones asociadas a cada símbolo $I(s_i)$ son $I(s_1)=2$, $I(s_2)=3$, $I(s_3)=I(s_4)=4$. La entropía de esta fuente (a la que llamaremos S_0) está dada por:

$$\begin{aligned}
 H(S_0) &= \sum_{i=1}^4 p(s_i) I(s_i) \\
 &= 0.5 \times 1 + 0.25 \times 2 + 0.125 \times 3 + 0.125 \times 3 \\
 &= 0.5 + 0.5 + 0.375 + 0.375 = 1.75
 \end{aligned} \tag{8}$$

Se puede notar que la cantidad mínima de bits requeridos para representar cuatro símbolos es dos. Si se elige una longitud fija para cada uno de los cuatro símbolos de nuestro alfabeto, podemos asignar un código como el que se muestra en la Tabla 11, en donde $\bar{L}=2$ y, de acuerdo con la ecuación (4), $H(S_0) \leq \log_2(4)=2$. Como se puede ver, efectivamente $1.75 < 2$.

Tabla 11. Código propuesto para el ejemplo del alfabeto hipotético:

Símbolo	Código
S1	00
S2	01
S3	10
S4	11

Se quisiera aproximar el valor de \bar{L} tanto como sea posible a $H(S_0)$. Para lograr esto, para empezar hay que considerar a los símbolos de menor probabilidad y asignarles un bit distintivo. En este caso, se empieza con s_3 y s_4 , asignándoles los bits 0 y 1 respectivamente. Ahora (y éste es el meollo del método) se puede considerar que el conjunto s_3+s_4 se “convierte” en un símbolo nuevo al que llamaremos s_{34} , y cuya probabilidad es la suma de sus componentes (en este caso $p(s_{34})=0.25$). Entonces se tiene un nuevo alfabeto con solamente tres símbolos (s_1 , s_2 y s_{34}) con probabilidades 0.5, 0.25 y 0.25, respectivamente. Repitiendo el procedimiento anterior, a s_{34} le corresponde un 0 y a s_2 un 1. Ahora componemos un nuevo símbolo s_{234} cuya probabilidad es 0.5. A s_{234} le asignamos un 0 y a s_1 un 1. Este procedimiento, aplicado de manera recursiva, arroja un árbol como el que se muestra en la Figura 12.17. De ese árbol se desprende, de inmediato, el código de Huffman, mostrado en la Tabla 12.

Tabla 12. Código de Huffman para el ejemplo del alfabeto hipotético:

Símbolo	Código de Huffman
s1	1
s2	01
s3	001
s4	000

Se puede ver que el árbol binario asigna longitudes lo más parecidas posibles a los subárboles cuyas probabilidades son parecidas. En el ejemplo de S_0 , podemos calcular \bar{L} usando la fórmula (9).

La aplicación del método de Huffman a P1 es un poco más laboriosa dado el número de símbolos que aparecen en dicho mensaje. En la Figura 12.18a se muestra la primera parte del proceso.

Símbolo	Probabilidad	Símbolo	Probabilidad	Símbolo	Probabilidad	Símbolo	Probabilidad
F	0.015625	F+G	0.031250	L+N	0.039063	F+G+S	0.062500
G	0.015625	S	0.031250	V+Y	0.046875	E	0.062500
L	0.015625	L+N	0.039063	I	0.046875	M	0.062500
N	0.023438	V+Y	0.046875	T	0.046875	R	0.070313
V	0.023438	I	0.046875	F+G+S	0.062500	A	0.078125
Y	0.023438	T	0.046875	E	0.062500	L+N+V+Y	0.085938
S	0.031250	E	0.062500	M	0.062500	I+T	0.093750
I	0.046875	M	0.062500	R	0.070313	D	0.093750
T	0.046875	R	0.070313	A	0.078125	H	0.093750
E	0.062500	A	0.078125	D	0.093750	K	0.132813
M	0.062500	D	0.093750	H	0.093750	Q	0.164063
R	0.070313	H	0.093750	K	0.132813		
A	0.078125	K	0.132813	Q	0.164063		
D	0.093750	Q	0.164063				
H	0.093750						
K	0.132813						
Q	0.164063						

Figura 12.18a. Árbol de Huffman para P1 (1a parte).

En la Figura 12.18a se puede ver cómo se van asociando los símbolos para formar símbolos compuestos cuyas probabilidades son mayores. En cada par de columnas símbolo/probabilidad, los símbolos han sido reordenados de acuerdo con su probabilidad. Cada columna de probabilidades suma 1.000.

La segunda parte del proceso se ilustra en la Figura 12.18b. Al final de esta fase, los 17 símbolos originales han sido aglutinados en solamente siete. La tercera parte de este proceso se puede observar en la Figura 12.18c. Al llegar a este punto los símbolos (y sus correspondientes probabilidades) se aglutinan en tres, los cuales luego se convierten en los dos que se muestran en la Tabla 13.

Tabla 13. Dos símbolos que se tienen en el penúltimo paso del método de Huffman aplicado al mensaje P1:

Símbolo	Probabilidad
I+T+D+H+F+G+S+E	0.406250
M+R+K+A+L+N+V+Y+Q	0.593750

El siguiente y último paso agrupa a estos dos pseudo-símbolos en la raíz del árbol para P1.

Símbolo	Probabilidad	Símbolo	Probabilidad	Símbolo	Probabilidad	Símbolo	Probabilidad
M	0.062500	A	0.078125	I+T	0.093750	H	0.093750
R	0.070313	L+N+V+Y	0.085938	D	0.093750	F+G+S+E	0.125000
A	0.078125	I+T	0.093750	H	0.093750	M+R	0.132813
L+N+V+Y	0.085938	D	0.093750	F+G+S+E	0.125000	K	0.132813
I+T	0.093750	H	0.093750	M+R	0.132813	A+L+N+V+Y	0.164063
D	0.093750	F+G+S+E	0.125000	K	0.132813	Q	0.164063
H	0.093750	M+R	0.132813	A+L+N+V+Y	0.164063	I+T+D	0.187500
F+G+S+E	0.125000	K	0.132813	Q	0.164063		
K	0.132813	Q	0.164063				
Q	0.164063						

Figura 12.18b. Árbol de Huffman para P1 (2a parte).

Símbolo	Probabilidad	Símbolo	Probabilidad	Símbolo	Probabilidad	Símbolo	Probabilidad
M+R	0.132813	A+L+N+V+Y	0.164063	I+T+D	0.187500	M+R+K	0.265625
K	0.164063	Q	0.164063	H+F+G+S+E	0.218750	A+L+N+V+Y+Q	0.328125
A+L+N+V+Y	0.164063	I+T+D	0.187500	M+R+K	0.265625	I+T+D+H+F+G+S+E	0.406250
Q	0.187500	H+F+G+S+E	0.218750	A+L+N+V+Y+Q	0.328125		
I+T+D	0.218750	M+R+K	0.265625				
H+F+G+S+E							

Figura 12.18c. Árbol de Huffman para P1 (3a parte).

De este proceso, análogamente a lo que se bosquejó en el caso de S0, se desprende el código de Huffman correspondiente para P1, mostrado en la Tabla 14. De esta tabla es fácil calcular la longitud promedio. Aquí $\bar{L} = 3.78125$, que es un poco mayor que $H(P1)$ (cuyo valor, anteriormente calculado, es 3.74377). Interesantemente, mientras que la longitud ASCII es de 856 bits, la longitud al codificar P1 con el método de Huffman es de solamente 484 bits, para una tasa de compresión de $856/484=1.7686$. Es decir, cada bit de nuestro mensaje codificado usando el método de Huffman equivale a casi 2 bits del mensaje original en ASCII.

Este es un ejemplo sencillo de la aplicación de la TEI a la compresión de datos sin pérdida. El lector interesado puede consultar, para obtener mayores detalles, por ejemplo a

(Burrows and Wheeler, 1994), (Cleary and Witten, 1984), (Elias, 1955), (Kuri and Galaviz, 2004) y (Langdon, 1984).

Tabla 14. Código de Huffman para P1:

Símbolo	Veces	Probabilidad	Código	Longitud	Long. Prom	Bits
F	2	0.015625	000000	6	0.093750	12
G	2	0.015625	000001	6	0.093750	12
L	2	0.015625	110000	6	0.093750	12
N	3	0.023438	110001	6	0.140625	18
V	3	0.023438	110010	6	0.140625	18
Y	3	0.023438	110011	6	0.140625	18
S	4	0.031250	00001	5	0.156250	20
I	6	0.046875	0100	4	0.187500	24
T	6	0.046875	0101	4	0.187500	24
E	8	0.062500	0001	4	0.250000	32
M	8	0.062500	1000	4	0.250000	32
R	9	0.070313	1001	4	0.281250	36
A	10	0.078125	1101	4	0.312500	40
D	12	0.093750	011	3	0.281250	36
H	12	0.093750	001	3	0.281250	36
K	17	0.132813	101	3	0.398438	51
Q	21	0.164063	111	3	0.492188	63

1.2.4.4. Compresión con pérdida.

En esta sección se considera el caso alternativo en el que es tolerable perder una parte de la información del mensaje para optimizar su transmisión y/o almacenamiento. Las técnicas de compresión con pérdida han cobrado primordial importancia para la reproducción de multimedia orientada al uso de los espectadores. En este caso se hace tolerable, por ejemplo, perder cierto detalle de las imágenes (lo que ha dado origen al estándar JPEG), cierta fidelidad en los sonidos (lo que ha dado origen al estándar MP3) o cierta fidelidad en los videos (lo que ha dado origen al estándar MPEG). Aquí se ejemplifica, de manera muy simple, la compresión de la imagen que se muestra en las Figuras 12.19a-b-c desde diferentes perspectivas. Para su mejor interpretación, la superficie ilustrada en dichas figuras se ha enmarcado en tres ejes coordenados, en donde la variable independiente X corresponde al plano más cercano al observador en la Figura 12.19b. En

esa misma figura, el eje cartesiano para la variable independiente Y corresponde al eje horizontal perpendicular al eje X . La variable dependiente Z corresponde al eje vertical.

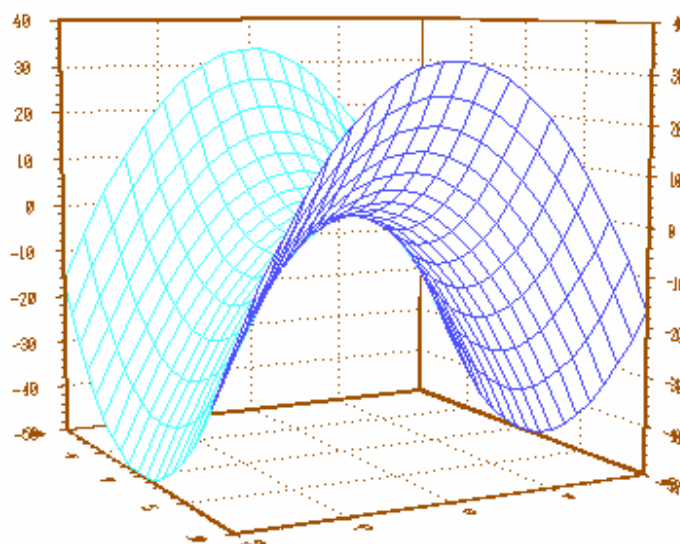


Figura 12.19a. Primera vista de un paraboloide hiperbólico.

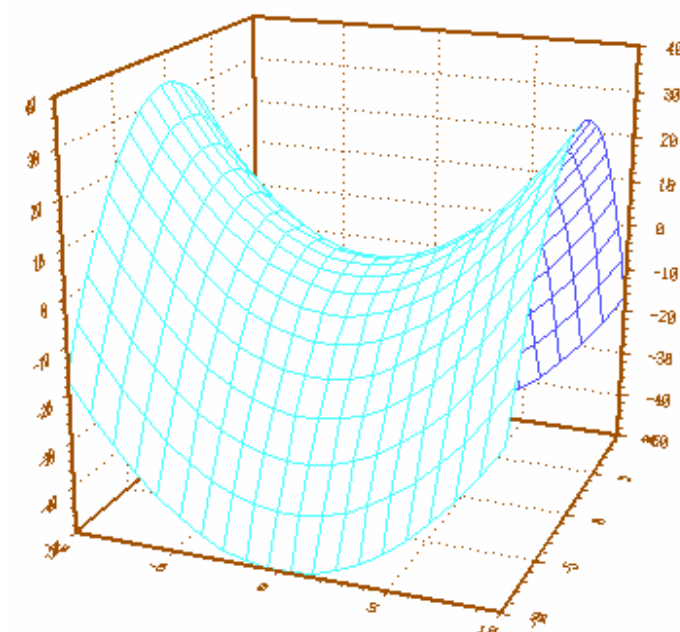


Figura 12.19b. Segunda vista de un paraboloide hiperbólico.

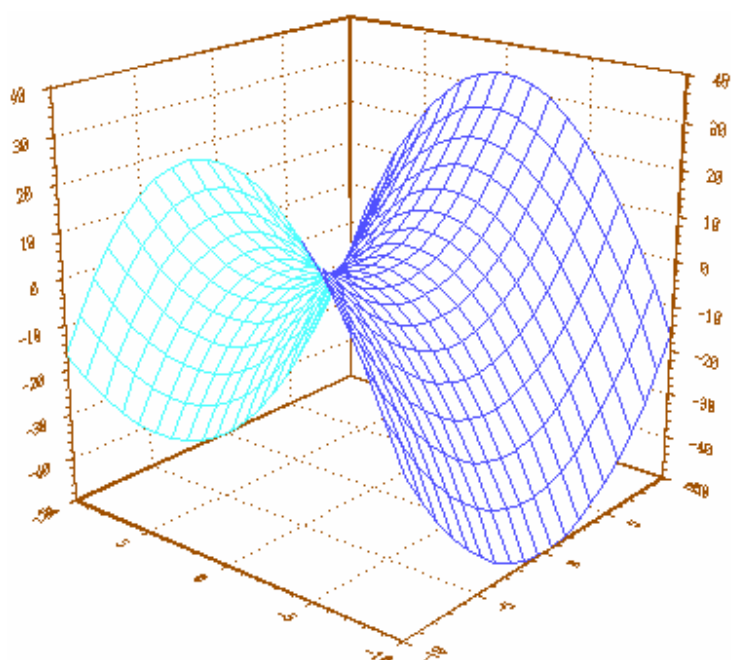


Figura 12.19c. Tercera vista de un paraboloide hiperbólico.

Las imágenes de las Figuras 12.19a-b-c se obtienen de la interpretación de los datos contenidos en la matriz que se muestra (parcialmente) en la Tabla 15.

Tabla 15. Vista parcial de los datos correspondientes a la superficie de las Figuras 12.19a-b-c:

XY	-10.000	-8.947	-7.895	-6.842	-5.789	-4.737	-3.684	-2.632	-1.579	-0.526	0.526		10.000
-10.000	-16.667	-6.694	2.170	9.926	16.574	22.115	26.547	29.871	32.087	33.195	33.195		-16.667
-8.947	-23.315	-13.343	-4.478	3.278	9.926	15.466	19.898	23.223	25.439	26.547	26.547		-23.315
-7.895	-29.224	-19.252	-10.388	-2.632	4.017	9.557	13.989	17.313	19.529	20.637	20.637		-29.224
-6.842	-34.395	-24.423	-15.559	-7.802	-1.154	4.386	8.818	12.142	14.358	15.466	15.466		-34.395
-5.789	-38.827	-28.855	-19.991	-12.235	-5.586	-0.046	4.386	7.710	9.926	11.034	11.034		-38.827
-4.737	-42.521	-32.549	-23.684	-15.928	-9.280	-3.740	0.693	4.017	6.233	7.341	7.341		-42.521
-3.684	-45.476	-35.503	-26.639	-18.883	-12.235	-6.694	-2.262	1.062	3.278	4.386	4.386		-45.476
-2.632	-47.692	-37.719	-28.855	-21.099	-14.451	-8.910	-4.478	-1.154	1.062	2.170	2.170		-47.692
-1.579	-49.169	-39.197	-30.332	-22.576	-15.928	-10.388	-5.956	-2.632	-0.416	0.693	0.693		-49.169
-0.526	-49.908	-39.935	-31.071	-23.315	-16.667	-11.127	-6.694	-3.370	-1.154	-0.046	-0.046		-49.908
0.526	-49.908	-39.935	-31.071	-23.315	-16.667	-11.127	-6.694	-3.370	-1.154	-0.046	-0.046		-49.908
1.579	-49.169	-39.197	-30.332	-22.576	-15.928	-10.388	-5.956	-2.632	-0.416	0.693	0.693		-49.169
2.632	-47.692	-37.719	-28.855	-21.099	-14.451	-8.910	-4.478	-1.154	1.062	2.170	2.170		-47.692
3.684	-45.476	-35.503	-26.639	-18.883	-12.235	-6.694	-2.262	1.062	3.278	4.386	4.386		-45.476
4.737	-42.521	-32.549	-23.684	-15.928	-9.280	-3.740	0.693	4.017	6.233	7.341	7.341		-42.521
5.789	-38.827	-28.855	-19.991	-12.235	-5.586	-0.046	4.386	7.710	9.926	11.034	11.034		-38.827
6.842	-34.395	-24.423	-15.559	-7.802	-1.154	4.386	8.818	12.142	14.358	15.466	15.466		-34.395
7.895	-29.224	-19.252	-10.388	-2.632	4.017	9.557	13.989	17.313	19.529	20.637	20.637		-29.224
8.947	-23.315	-13.343	-4.478	3.278	9.926	15.466	19.898	23.223	25.439	26.547	26.547		-23.315
10.000	-16.667	-6.694	2.170	9.926	16.574	22.115	26.547	29.871	32.087	33.195	33.195		-16.667

En la Tabla 15, la fila superior corresponde a los valores de X , la columna izquierda a los valores de Y y los elementos intermedios a los valores de Z . Se puede ver que hay 20 valores de X y otros tantos de Y ; en tanto que hay 400 valores para Z . Siguiendo las convenciones anteriores, vemos que se requiere de $440 \times 32 = 14080$ bits para almacenar esta información. Ahora encontraremos los coeficientes de la ecuación (7) correspondientes a los datos de la Tabla 15. Elegimos una potencia máxima de grado 2, de manera que el aproximador polinomial es de la forma:

$$f(x,y) = c_{00} + c_{01}y + c_{02}y^2 + c_{10}x + c_{11}xy + c_{12}xy^2 + c_{20}x^2 + c_{21}x^2y + c_{22}x^2y^2 \quad (10)$$

Aplicando técnicas de aproximación multivariada (Cheney, 1982), obtenemos los valores de los coeficientes $c_{00}, c_{01}, \dots, c_{22}$ que hacen que la ecuación (10) se ajuste a los datos de la Tabla 15 con el menor error posible. En general no es posible ajustar los datos sin error usando un número reducido de coeficientes. Podríamos ajustarlos todos con un coeficiente para cada elemento, pero entonces no lograríamos una nueva expresión más compacta (que es nuestro objetivo). Los valores de $c_{00}, c_{01}, \dots, c_{22}$ resultantes se muestran en la Tabla 16.

Tabla 16. Coeficientes de ajuste para los datos de la Tabla 15:

C00	-0.0000107400
C01	0.0000019264
C02	-0.4999998367
C10	0.0000018057
C11	-0.0000017711
C12	-0.0000003175
C20	0.3333329745
C21	-0.0000002417
C22	-0.0000000254

Se puede ver en la Tabla 16 que algunos de los valores de los coeficientes son muy pequeños, en tanto que el coeficiente c_{02} , asociado a la segunda potencia de Y , y c_{20} , asociado a la segunda potencia de X , resultan ser muy significativos. Ahora bien, para representar los nueve coeficientes solamente requerimos de $9 \times 32 = 288$ bits. La tasa de compresión, en este caso, es de

14,080/288=48.8889. En otras palabras, ¡cada bit de nuestro mensaje comprimido corresponde a casi 50 bits del mensaje original! Se puede contrastar este resultado con el anterior (sin pérdida), cuya tasa de compresión estaba acotada por la entropía. Pero, a cambio de lograr esta impresionante eficiencia de representación ¿qué se ha perdido? Resulta que el error máximo de aproximación es, en este ejemplo de sólo 0.0009834002. ¡Perdemos, cuando mucho, cerca de una milésima de precisión! Este error, en casos de mediciones científicas o para efectos de manejo de transferencias bancarias, por ejemplo, es inaceptable. Pero para observar presentación de una imagen es un error perfectamente admisible. El resultado de recalculer los datos a partir de los coeficientes anteriores se muestra en la Tabla 17.

Tabla 17. Vista parcial de los datos correspondientes a los coeficientes de las Figuras 12.19a-b-c:

XY	-10.000	-8.947	-7.895	-6.842	-5.789	-4.737	-3.684	-2.632	-1.579	-0.526	0.526			10.000
-10.000	-16.667	-6.694	2.170	9.926	16.574	22.114	26.547	29.871	32.087	33.195	33.195			-16.667
-8.947	-23.315	-13.343	-4.478	3.278	9.926	15.466	19.898	23.222	25.439	26.547	26.547			-23.315
-7.895	-29.224	-19.252	-10.388	-2.632	4.017	9.557	13.989	17.313	19.529	20.637	20.637			-29.224
-6.842	-34.395	-24.423	-15.559	-7.802	-1.154	4.386	8.818	12.142	14.358	15.466	15.466			-34.395
-5.789	-38.827	-28.855	-19.991	-12.235	-5.586	-0.046	4.386	7.710	9.926	11.034	11.034			-38.827
-4.737	-42.521	-32.548	-23.684	-15.928	-9.280	-3.740	0.692	4.017	6.233	7.341	7.341			-42.521
-3.684	-45.476	-35.503	-26.639	-18.883	-12.235	-6.694	-2.262	1.062	3.278	4.386	4.386			-45.476
-2.632	-47.692	-37.719	-28.855	-21.099	-14.451	-8.910	-4.478	-1.154	1.062	2.170	2.170			-47.692
-1.579	-49.169	-39.197	-30.332	-22.576	-15.928	-10.388	-5.956	-2.632	-0.416	0.693	0.693			-49.169
-0.526	-49.908	-39.935	-31.071	-23.315	-16.667	-11.127	-6.694	-3.370	-1.154	-0.046	-0.046			-49.908
0.526	-49.908	-39.935	-31.071	-23.315	-16.667	-11.127	-6.694	-3.370	-1.154	-0.046	-0.046			-49.908
1.579	-49.169	-39.197	-30.332	-22.576	-15.928	-10.388	-5.956	-2.632	-0.416	0.693	0.693			-49.169
2.632	-47.692	-37.719	-28.855	-21.099	-14.451	-8.910	-4.478	-1.154	1.062	2.170	2.170			-47.692
3.684	-45.476	-35.503	-26.639	-18.883	-12.235	-6.694	-2.262	1.062	3.278	4.386	4.386			-45.476
4.737	-42.521	-32.549	-23.684	-15.928	-9.280	-3.740	0.693	4.017	6.233	7.341	7.341			-42.521
5.789	-38.827	-28.855	-19.991	-12.235	-5.586	-0.046	4.386	7.710	9.926	11.034	11.034			-38.828
6.842	-34.395	-24.423	-15.559	-7.802	-1.154	4.386	8.818	12.142	14.358	15.466	15.466			-34.396
7.895	-29.225	-19.252	-10.388	-2.632	4.017	9.557	13.989	17.313	19.529	20.637	20.637			-29.225
8.947	-23.315	-13.343	-4.478	3.278	9.926	15.466	19.898	23.223	25.439	26.547	26.547			-23.316
10.000	-16.667	-6.694	2.170	9.926	16.574	22.115	26.547	29.871	32.087	33.195	33.195			-16.668

Como se puede apreciar, las diferencias entre la Tabla 15 y la Tabla 17 no son significativas si se usa una precisión de tres cifras decimales, como se han desplegado los números en dichas tablas. La prueba final, por supuesto, depende del observador. A ese efecto presentamos las imágenes derivadas de graficar la Tabla 17 en las Figuras 12.20a-b-c.

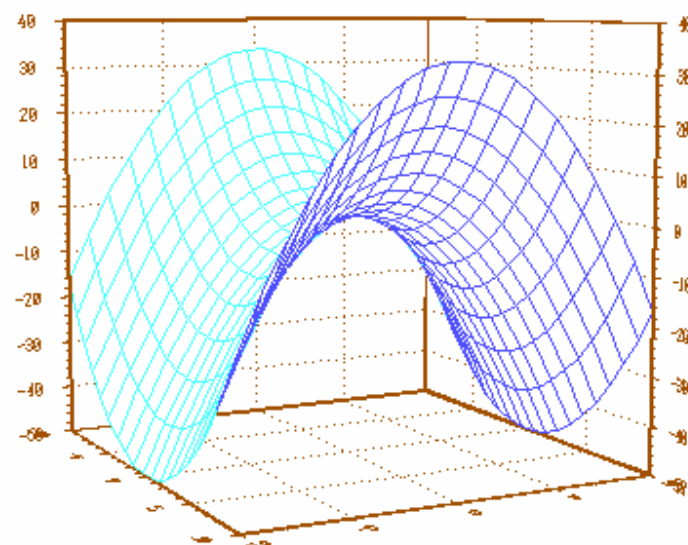


Figura 12.20a. Primera vista de una imagen generada después de una compresión con pérdida.

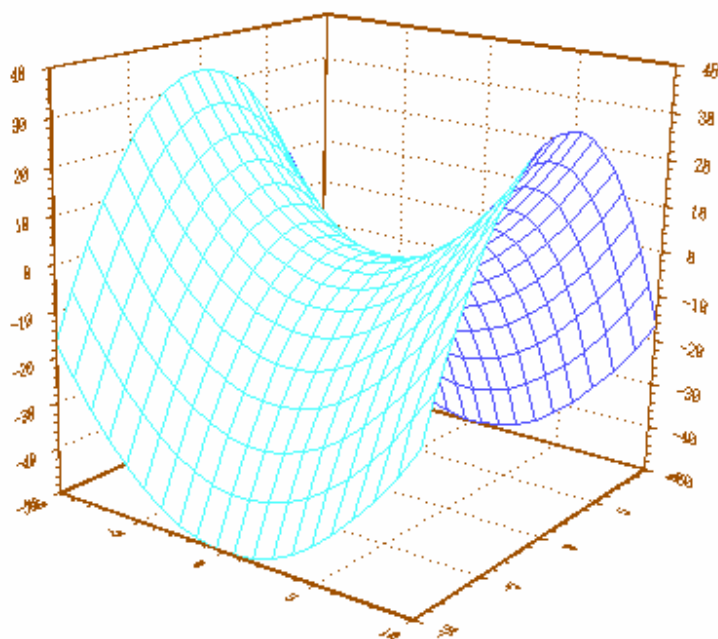


Figura 12.20b. Segunda vista de una imagen generada después de una compresión con pérdida.

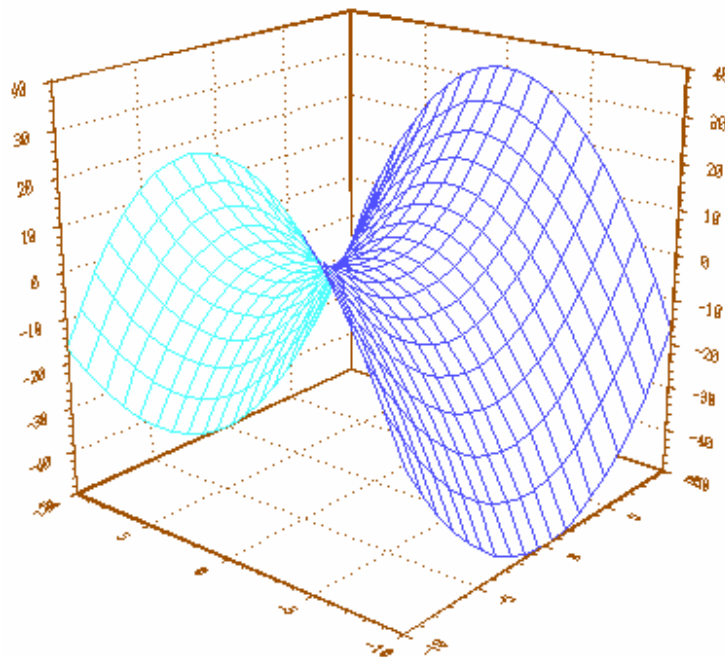


Figura 12.20c. Tercera vista de una imagen generada después de una compresión con pérdida.

Parecería que las imágenes son idénticas, ¿no? Este es el principio general en el que se basan las técnicas de compresión con pérdida: reproducir un conjunto de datos de forma que sean lo suficientemente parecidos a los datos originales. Aunque las técnicas mencionadas anteriormente para los estándares de compresión con pérdida no tienen la misma estructura algorítmica que nuestras aproximaciones polinomiales, la idea es exactamente la misma. Nosotros pretendemos ilustrar, con este ejemplo sencillo, una posible aplicación de la versión algorítmica de la teoría de la información. El lector interesado en los estándares actuales de compresión de datos con pérdida puede consultar (JPEG Committee, 2006), (askMP3, 2006) y (MPEG standards, 2006).

1.2.5. Un enfoque híbrido.

Algo que no se ha mencionado, pero que el lector posiblemente ha advertido, es que nuestros cálculos de compresión, tanto para el caso sin pérdida como para el caso con pérdida, han pasado por alto que la mera transmisión de los datos es insuficiente para regenerar el mensaje

que Tex le manda a Rex. Esto se debe a que, en el caso de los códigos de Huffman, Rex no sabe los códigos para ese caso específico, mientras que en el caso de los coeficientes polinomiales, no sabe los grados de los coeficientes de antemano. Podríamos establecer convenciones, pero ya hemos visto que, en general, no sabemos con anticipación la mejor codificación de Huffman (porque depende del mensaje) o, en su caso, el grado máximo del polinomio (porque depende de qué tanta información estamos dispuestos a perder).

Lo que está implícito es que, para que Rex pueda reproducir (entender) el mensaje que le envió Tex, hay que enviarle no solamente los bits que codifican el mensaje, sino también las especificaciones de codificación. A este conjunto de especificaciones se les denomina el **modelo del mensaje**. Así pues, lo que realmente queremos minimizar es la suma del mensaje con el modelo. Todos los algoritmos de compresión, en la práctica, siguen esta convención y a esto se le llama el **principio de descripción mínima** que fue postulado, originalmente, en (Rissanen, 1978).

Para ejemplificar, si tomamos el caso del envío del mensaje P1 (véase la subsubsubsección “Compresión sin pérdida” arriba) con códigos de Huffman, el modelo tendría que incluir los códigos de las letras en él contenidas, así como un indicador de qué código corresponde a qué letra. En la siguiente secuencia (que se tendría que enviar junto con el mensaje) cada carácter va seguido de su código:

“:F000000:G000001:L110000:N110001:V110010:Y110011:S00001:I0100:T0101:E0001:M1000:R1001:A1101:D011:H001:K101:Q111”

Claro que hay que predefinir en qué código se van a representar los caracteres en este preámbulo (llamado así porque generalmente se transmite previo a, o al inicio de, la transmisión del mensaje). Si se decide por el código ASCII, requerimos de $7 \times 17 = 119$ bits adicionales a los 77 que representan las combinaciones de 0s y 1s del código Huffman que se va a usar para

representar los caracteres en el mensaje (más los bits correspondientes al separador “:”) para indicar a qué letra corresponde cada código. Rex interpretará los primeros 7 bits como la definición del carácter y los siguientes (hasta llegar a “:”) como su código Huffman. ¿Cómo se representa el separador “:”? Usando un conjunto de bits que no aparezcan como parte de ninguno de los códigos de Huffman. Por cierto, ninguna representación de ningún carácter en código Huffman está contenida como prefijo de ninguna otra representación. La longitud de representación de “:” debe decidirse de antemano y forma parte del modelo. Puesto que solamente se tienen 17 símbolos, el uso de ASCII es poco económico, ya que solamente se requieren cinco bits para expresar 17 combinaciones. De hecho, por razones que se explican más adelante, se incluyen en nuestro alfabeto, además de los 17 símbolos ya mencionados, los caracteres “C”, “P” y “W”. De esta manera se puede convenir en enviar los códigos usando cinco bits y no los siete del código ASCII. Así, los bits adicionales necesarios para codificar las letras se disminuyen de 119 a $5 \times 17 = 85$. Se pueden obtener todavía mejores resultados si se decide enviar los caracteres alfabéticamente ordenados. Así no es necesario ni siquiera incluir dichos caracteres. Las 20 letras de P1 quedarían ordenadas como se ve en la Tabla 18.

Tabla 18. Ordenamiento de los caracteres del mensaje P1:

Orden	Símbolo	Orden	Símbolo	Orden	Símbolo
1	A	8	I	15	R
2	C	9	K	16	S
3	D	10	L	17	T
4	E	11	M	18	V
5	F	12	N	19	W
6	G	13	P	20	Y
7	H	14	Q		

Con esta convención, el modelo de P1 se convierte en la siguiente secuencia:

“1101::011:0001:000000:000001:001:0100:101:110000:1000:110001::111:1001:0000
1:0101:110010::110011”

La ausencia de alguna de las 20 letras se detecta fácilmente porque aparecen dos separadores “::” consecutivos en la secuencia (en los lugares en los que debieran aparecer los códigos de las letras ausentes). Así sabemos que “C”, “P” y “W” no aparecen en este mensaje. ¡Se han ahorrado 119 bits al adoptar esta convención, puesto que sólo se envían los códigos y los separadores (en lugar de enviar los símbolos junto con sus códigos y además los separadores)!

Como se ve del ejemplo anterior, además del mensaje es importante decidir la mejor forma de mandar el modelo. Esto implica un compromiso para lograr que, de forma simultánea, el mensaje y el modelo se codifiquen de la mejor manera posible. En esto consiste el principio de descripción mínima (criterio de Rissanen), arriba mencionado, conocido también como el principio MDL (del inglés **minimum description length**). Para satisfacer este principio en la práctica, se usa una especie de híbrido que considera aspectos de la TEI y la TAI de manera simultánea. Sin entrar en detalle (el lector interesado puede consultar (Cormack and Horspool, 2001), (Einarsson, 1991), (Witten, Neal, and Cleary, 1987), (Yu, 1996) o (Ziv and Lempel, 1977), entre otros), las técnicas de compresión sin pérdida se pueden dividir, a grandes rasgos, en técnicas estadísticas (como la constituida por los códigos de Huffman) y técnicas de diccionario (que son las que se usan más comúnmente en software comercial como WinZip^(R) (WinZip, 2006)). En ambos casos lo que se desea es utilizar el principio MDL para optimizar el tamaño total de la información que se envía o almacena.

1.2.5.1. Compresión sin pérdida basada en patrones.

Una de las limitantes de la TEI es que los símbolos de un mensaje se definen de manera arbitraria. Usar bytes o secuencias de bytes (palabras, por ejemplo) para representar los símbolos es muy natural, pero no es necesariamente lo mejor. Una alternativa desarrollada recientemente consiste en no definir la estructura de los símbolos, sino permitir que una computadora analice el texto y lo haga por nosotros. Una vez determinado el conjunto óptimo de

esos símbolos (que se llaman **metasímbolos**, por estar compuestos de colecciones de símbolos, no necesariamente adyacentes dentro del mensaje, agrupados en patrones que se repiten), se pueden usar las técnicas discutidas anteriormente para compactar los datos de la manera más eficiente de acuerdo con el principio MDL.

Para ilustrar la idea, regresemos al mensaje P1 que se ha acomodado en una matriz de 16x8 como se muestra en la Tabla 19. Los símbolos de P1 corren de izquierda a derecha y de arriba a abajo, como es usual. No es evidente, pero en esta tabla existen seis patrones distintos. Los primeros cinco se repiten una o más veces. El otro (el relleno) solamente se encuentra una vez en el mensaje.

Tabla 19. Mensaje P1 representado de forma matricial:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
M	S	M	Q	E	T	Q	V	Q	H	F	R	K	D	R	Q	1
A	R	R	A	E	R	A	Q	A	K	D	N	R	F	Q	A	2
H	Q	Q	T	I	Q	M	R	D	M	E	K	Q	V	K	K	3
Q	Y	K	D	H	H	D	Q	M	K	E	S	E	T	H	V	4
D	N	I	K	K	D	D	K	H	A	I	E	E	D	Q	M	5
D	K	D	N	R	R	S	Y	K	S	H	I	I	R	H	L	6
Q	M	H	T	Q	Q	H	Q	Q	Y	M	Q	H	Q	D	Q	7
A	K	K	E	K	T	H	G	A	I	K	G	A	L	K	A	8

El primer metasímbolo (ms_1) consiste de 12 símbolos (letras, en este caso) y aparece tres veces. La posición del primer símbolo (“M”) de la primera instancia de ms_1 está en (1,1) (véase la Tabla 20). Los primeros símbolos de la segunda y tercera instancias están, respectivamente, en (7,3) y (9,4).

Tabla 20. Primera instancia del primer metasímbolo ms_1 que aparece en el mensaje mostrado en la Tabla 19:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
M				E			V						D			1
				E						D	N					2
				I					M							3
	Y			H									T			4
																5
																6
																7
																8

Como se ve, un metasímbolo consiste de un conjunto de símbolos (elementos definidos) ubicados dentro de una estructura fija en donde los símbolos tienen valores definidos pero el resto de los elementos de la estructura no los tienen. Por ejemplo, nótese que entre la M y la E iniciales de ms_1 hay tres espacios. Que los espacios estén vacíos no significa que no tengan contenido en el mensaje, sino que el contenido está indefinido en la estructura del metasímbolo. Se puede imaginar que el metasímbolo está impreso sobre una superficie transparente, formando una máscara. Si esta máscara se desliza sobre el mensaje, habrá momentos de coincidencia con los símbolos del mensaje. Para ms_1 los puntos de coincidencia están en (1,1), (7,3) y (9,4), como ya se mencionó. Es posible identificar un segundo metasímbolo (ms_2), como se muestra en la Tabla 21.

Tabla 21. Primera instancia de un segundo metasímbolo ms_2 que aparece en el mensaje mostrado en la Tabla 19:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
																1
																2
H								D								3
Q											S					4
								H								5
D											I				L	6
																7
																8

El metasímbolo ms_2 consiste de ocho símbolos y aparece dos veces. Su posición inicial es (1,3) en la primera instancia y (15,4) en la segunda instancia. El metasímbolo ms_3 consiste de tres símbolos y aparece nueve veces, en las posiciones (15,1), (2, 2), (3, 2), (6, 2), (13, 2), (8, 3), (5, 6), (6, 6) y (14, 6). El metasímbolo ms_4 consta de dos símbolos y aparece siete

veces, en las posiciones (4,1), (7, 1), (9, 1), (16, 1), (1, 7), (9, 7) y (16, 7). El metasímbolo ms_5 consta de tres símbolos y aparece dos veces, en las posiciones (15,6) y (3,7). Las primeras instancias de ms_3 , ms_4 y ms_5 en P1 se muestran en la Tabla 22.

Tabla 22. Primeras instancias de los metasímbolos ms_3 , ms_4 y ms_5 que aparecen en el mensaje mostrado en la Tabla 19:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
			Q											R		1
			A											Q		2
											K					3
																4
																5
														H		6
							Q									7
							G									8

Los símbolos de todas las instancias de ms_1 a ms_5 comprenden 99 de los 128 símbolos de P1. Los restantes conforman el relleno, que se muestra en la Tabla 23.

Por comodidad, se designará a los metasímbolos (sin incluir el relleno) con las letras griegas α , β , γ , δ y ε , respectivamente. La representación de P1, usando únicamente metasímbolos, es: “ $\alpha\delta\delta\delta\gamma\delta\gamma\gamma\gamma\beta\alpha\gamma\alpha\beta\gamma\gamma\varepsilon\delta\varepsilon\delta\delta$ ”. Se puede comparar el mensaje original P1 con su nueva versión metasimbólica. En vez de 128 símbolos (cuyos contenidos y “estructura” se definieron arbitrariamente), restan solamente 23 metasímbolos. Si se definiera un catálogo de metasímbolos, bastaría con enumerar 23 de ellos para reproducir el mensaje original completo de manera exacta. Este es otro ejemplo de compresión sin pérdida.

Tabla 23. El metasímbolo “relleno” que aparece en el mensaje mostrado en la Tabla 19:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
	S	M			T				H	F	R	K				1
A							Q		K				F			2
			T													3
					H	D										4
		I	K				K		A		E		D			5
	K					S		K								6
										M						7
			E	K		H						A		K		8

Un mapa de P1 expresado por los metasímbolos dentro de la matriz se muestra en la Tabla 24. Los metasímbolos incluidos en dicha tabla se muestran en la posición que ocupa el primer símbolo del metasímbolo en cuestión.

Tabla 24. Mensaje mostrado en la Tabla 19 representado por sus metasímbolos:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
α			δ			δ		δ						γ	δ	1
	γ	γ			γ							γ				2
β						α	γ									3
								α						β		4
																5
				γ	γ								γ	ε		6
δ		ε						δ							δ	7
																8

De no contar con el catálogo de metasímbolos mencionado, el principio MDL nos obliga a enviar, junto con el mensaje, el modelo del mismo, como ya se comentó anteriormente. El lector interesado en los detalles puede consultar (Kuri and Ortiz, 2005).

El método bosquejado combina ambas teorías de la información, la estadística y la algorítmica. Por una parte el método encuentra el conjunto óptimo de metasímbolos, que es equivalente a encontrar la complejidad de Kolmogorov, ya que un mensaje expresado como la colección óptima de metasímbolos no puede ser representado de manera más compacta. Por otra parte el método aprovecha las posibilidades de codificación derivadas de la teoría de Shannon. El problema básico radica, precisamente, en que la CK no es computable y, por lo tanto, en la práctica debemos conformarnos con expresiones aproximadas que, sin embargo, compiten

favorablemente con otras alternativas más usuales. Para una discusión más amplia de la compresión de datos usando patrones, el lector interesado puede consultar (Kuri Morales, 2004) y (Kuri Morales, Galaviz Casas, and Herrera Alcántara, 2005).

1.2.6. Consideraciones finales.

Para terminar este capítulo ofrecemos un ejemplo de aplicación de la teoría de la información en las ciencias biológicas. En párrafos anteriores se mencionó que sería conveniente incluir las letras “C”, “P” y “W” en el alfabeto del mensaje P1. La razón es que cada una de las letras de este alfabeto corresponde a uno de los 20 aminoácidos comunes de las células de los seres vivos. La información total que describe a los seres vivos se encuentra codificada en el ADN que está presente en el núcleo de cada célula. El ADN es una larga secuencia de **bases** (compuestos químicos llamados adenina, guanina, citosina y timina) que, al ser interpretadas en grupos de tres se pueden interpretar como secuencias de, básicamente, 20 aminoácidos, a cada uno de los cuales se le ha dado un nombre. Para facilidad de los estudiosos de la microbiología (y de otras disciplinas del conocimiento), cada aminoácido se representa con una letra. El alfabeto de P1 es, justamente, el de los aminoácidos. P1, bajo esta luz, es una cadena de instrucciones para la operación de un ser vivo, es decir lo que se conoce como una **proteína**. Las proteínas son los programas que gobiernan el funcionamiento de todas y cada una de las funciones de la vida. El ejercicio de reexpresión de P1 en aminoácidos se convierte, por tanto, en la búsqueda de los patrones que regulan el funcionamiento de las células. De estos patrones depende, en última instancia, la vida misma. A la rama de la computación que se encarga del análisis de las proteínas se le denomina **proteómica computacional**. En los seres vivos (aún en los más simples) hay varios miles de proteínas. En el ser humano, por ejemplo, se han identificado cerca de 40000 proteínas distintas. Cada proteína tiene una función específica y el hecho de encontrar patrones

(metasímbolos) similares en conjuntos de ellas es un indicativo de las funciones que la proteína regula. Este problema es tan importante en las ciencias biológicas que ha dado origen a la disciplina conocida como Bioinformática. La determinación de secuencias de aminoácidos y nucleótidos equivale a encontrar los metasímbolos adecuados en las células. Es una de las ramas de la ciencia que ha recibido mayor atención a raíz del proyecto del genoma humano (véase, por ejemplo, (Gibbs and McIntyre, 1970)). Como se puede ver, hay aplicaciones directas de la teoría de la información a la biología molecular. Para una introducción a este interesante tema, el lector interesado puede ver, por ejemplo, (Hofstadter, 1979).

De lo anteriormente discutido se desprende que la teoría de la información nos provee de un marco teórico con el cual analizar las diversas manifestaciones del conocimiento a través de su codificación y posterior almacenamiento en los sistemas de cómputo digital. Nos permite conocer los límites prácticos a los que se puede aspirar al tratar de codificar un mensaje de la manera más eficiente. En resumen, nos da la posibilidad de entender qué tan útil puede ser un conjunto de datos en base a una medida objetiva y clara. La cantidad de aplicaciones de esta teoría es virtualmente ilimitada. La información contenida en casi cualquier objeto de conocimiento, gracias a la teoría de la información, puede ser medida y cuantificada. Con ello podemos responder nuestra pregunta inicial, ¿qué tan útil es lo que estoy leyendo?

Preguntas de repaso y ejercicios.

1. Un mensaje se envía a un destinatario desconocido y debe poder ser decodificado sin que el destinatario posea información alguna acerca del código, el formato u otro elemento componente del mensaje. ¿Cómo plantearía la solución problema? Vea, por ejemplo, “Murmullos de la Tierra”, de Carl Sagan (Sagan, 1978).

2. Escriba un programa que mida la información en un conjunto arbitrario de datos, tomando como probabilidad de un símbolo la proporción de ocurrencias de éste dentro del total de ellos. Elija $n=16$. Ahora repita lo mismo, pero haciendo $n=8$. Discuta las diferencias de ambas elecciones.

3. Suponga la siguiente tabla de probabilidades para los símbolos de un alfabeto.

s1	0.50000
s2	0.12500
s3	0.03125
s4	0.25000
s5	0.03125
s6	0.06250

Encuentre la entropía, el código de Huffman y la longitud promedio respectiva. Comente sus resultados.

4. Para codificar imágenes se pueden usar los denominados “canales de color”. ¿Qué son y qué aplicaciones tienen?

5. Si grabamos la ejecución de una orquesta sinfónica durante 50 minutos, ¿cuántos bits deberemos almacenar suponiendo que se sigan los estándares mencionados en el texto? Suponga que una página típica de un libro consta de 60 renglones y 80 columnas, que cada página se llena con caracteres en un 90% de las veces y que un libro típico consta de 250 páginas. ¿A cuántos de estos libros equivale la información grabada (de la orquesta sinfónica)?

6. Explique el concepto de una máquina de Turing.

7. Suponga que tiene la descripción de un objeto en cinco dimensiones como una colección de vectores de cinco coordenadas cada uno (cuatro variables independientes y una dependiente). Suponga que va a hacer una representación polinomial de la variable dependiente como función de las otras cuatro variables. Suponga que utiliza $g_1=g_2=g_3=g_4=6$. ¿Cuántos valores requeriría para obtener la descripción polinomial de este objeto?

8. ¿Cuántas dimensiones requiere para expresar una película como un objeto susceptible de ser aproximado polinomialmente? Describa sus convenciones.

9. En un mensaje de 128 símbolos (tal como P1) y un alfabeto de 20 letras, ¿cuántos metasímbolos diferentes es posible identificar?

10. En el texto se afirma que “encontrar el conjunto óptimo de metasímbolos es equivalente a encontrar la complejidad de Kolmogorov, ya que un mensaje expresado como la colección óptima de metasímbolos no puede ser representado de manera más compacta”. Explique por qué. ¿Qué significa “la colección óptima”, en este contexto?

11. Identifique cada una de las instancias de ms_1 en la Tabla 19. ¿Qué porcentaje del total del mensaje abarca la suma de las instancias de ms_1 ?

12. Para completar la descripción de un mensaje es necesario definirlo como una colección de metasímbolos. Para cada uno de ellos hay que estipular su estructura y su contenido. Sin embargo, al final hay que completar la descripción del mensaje con el relleno. Este pseudo-metasímbolo queda definido de manera absoluta con solamente estipular sus contenidos. ¿Por qué sí o por qué no (es cierta esta aseveración)?

Referencias

(askMP3, 2006) *MP3 standards home page*, www.mpeg.org/MPEG/mp3.html.

(Burrows and Wheeler, 1994) Burrows, M. and Wheeler, D., “A block-sorting lossless data compression algorithm”, *Research report 124*, Digital Systems Research Center, May 1994.

(Chaitin, 2004) Chaitin, G.J., “Algorithmic information theory”, Cambridge University Press, 2004.

(Cheney, 1982) Cheney, E. W., *Introduction to approximation theory (2nd ed.)*, New York: Chelsea, 1982.

(Cleary and Witten, 1984) Cleary, J.G. and Witten, I., “Data compression using adaptive coding and partial string matching”, *IEEE transactions on communications*, Vol. 32, No. 4, pp. 396-402, 1984.

- (Cormack and Horspool, 2001) Cormack, G. and Horspool, R., "Data compression using dynamic Markov modeling", *The computer journal*, Vol. 30, No. 6, pp. 541-550, 1986.
- (Einarsson, 1991) Einarsson, G., "An improved implementation of predictive coding compression", *IEEE transactions on communications*, Vol. 39, No. 2, pp. 169-171, 1991.
- (Elias, 1955) Elias, P., "Predictive coding", *IRE Transactions on Information Theory*, Vol 1, pp. 16-33, 1955.
- (Gibbs and McIntyre, 1970) Gibbs, A.J. and McIntyre, G.A., "The diagram: a method for comparing sequences. Its use with aminoacids and nucleotide sequences", *European journal on biochemistry*, Vol. 16, pp 1-11, 1976.
- (Hofstadter, 1979) Hofstadter, D., *Gödel, Escher and Bach: an eternal golden braid*, Basic Books, pp. 504-526, 1979.
- (Huffman, 1952) Huffman, D., "A method for the construction of minimum redundancy codes", *Proceedings of IRE*, Vol. 40, No. 9, pp. 1098-1101, 1952.
- (JPEG Committee, 2006) JPEG Committee, www.jpeg.org, 2006.
- (Kolmogorov, 1965) Kolmogorov, A. N., "Three approaches to the quantitative definition of information", *Problems on information transmission*, Vol. 1, No. 1, pp. 1-7, 1965.
- (Kuri and Galaviz, 2004) Kuri, A. and Galaviz, J., "Pattern-based data compression", *Proceedings of MICAI 2004: advances in artificial intelligence*, Springer-Verlag, LNAI 2972, pp. 1-10, 2004.
- (Kuri Morales, 2004) Kuri Morales, A., "Pattern based lossless data compression", *WSEAS Transactions on Communications*, Issue 1, Vol. 3, pp. 42-50, WSEAS Press, ISBN 1109-2742, 01/01/2004.
- (Kuri and Ortiz, 2005) Kuri, A. and Ortiz, M., "A new approach to sequence representation of proteins in bioinformatics", *Proceedings of the 4th mexican international congress on artificial intelligence*, LNAI 3789, Springer Verlag, 2005.
- (Kuri Morales, Galaviz, and Herrera Alcántara, 2005) Kuri Morales, A., Galaviz Casas, J., and Herrera Alcántara, O., "Practical estimation of Kolmogorov complexity using highly efficient compression algorithms", *Advances in artificial intelligence applications*, pp. 193-201, Politécnico, (Editor(s)): Gelbukh, A., Monroy, R., ISSN 165-9899, 15/11/2005.
- (Langdon, 1984) Langdon, G., "An introduction to arithmetic coding", *IBM journal of research and development*, Vol. 28, No. 2, pp. 135-149, March 1984.
- (Li and Vitányi, 1997) Li, M. and Vitányi, P., *An introduction to Kolmogorov complexity and its applications* (2nd. Ed.), Springer, 1997.
- (MPEG standards, 2006) International Organisation for Standardisation ISO/ IEC JTC1/ SC29/ WG11, *Coding of moving pictures and audio*, www.chiariglione.org/mpeg/standards/mpeg-4/mpeg-4.htm, 2006.
- (Rissanen, 1978) Rissanen, J., "Modeling by shortest data description", *Automatica*, Vol. 14, pp. 465-471, 1978.
- (Sagan, 1978) Sagan, C., *Murmurs of the Earth: the Voyager interstellar record*, Random House, 1978.

(Shampine, Allen, and Pruess, 1997) Shampine, F., Allen, R., and Pruess, S. *Fundamentals of numerical computing*, John Wiley & Sons, 1997.

(Shannon, 1948) Shannon, Claude E., "A mathematical theory of communication", *The Bell system technical journal*, Vol. 27, July: pp. 379-423, October: pp. 623-656, 1948.

(Scheid, 1968) Scheid, F., "Numerical analysis", *Schaum's Outline Series*, McGraw-Hill Book Company, 1968.

(Wikipedia, 2006) 4'33, <http://en.wikipedia.org/wiki/4'33%22>.

(WinZip, 2006) Corporativo WinZip, *WinZip: the zip file utility for Windows*, www.winzip.com, 2006.

(Witten, Neal, and Cleary, 1987) Witten, I, Neal R., and Cleary, J., "Arithmetic coding for data compression", *Communications of the ACM*, Vol. 30, No. 6, pp. 520-540, 1987.

(Yu, 1996) Yu, T., "Dynamic Markov Compression", *Dr. Dobbs's Journal*, pp. 30-32, 1996.

(Ziv and Lempel, 1977) Ziv, J., and Lempel, A., "A universal algorithm for sequential data compression", *IEEE transactions on information theory*, Vol. 23, No. 3, pp. 337-343, 1977.