Recibido: martes, 27 marzo 2007

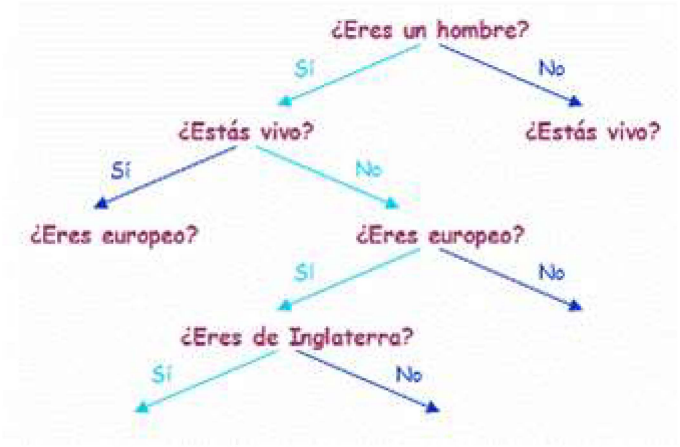
ciencia :: matematicalia
revista digital de divulgación matemática
Vol. 3, no. 2 (abr. 2007)

Texto, bytes y cintas de vídeo (*)

Oliver Johnson
Departamento de Matemáticas
Universidad de Bristol
e-mail: O.Johnson@bristol.ac.uk
página web: http://www.stats.bris.ac.uk/~maotj

Veinte preguntas

¿Recuerdas el juego de "Veinte preguntas"? Un jugador piensa en un personaje famoso y los otros tienen que averiguar de quién se trata, preguntando no más de 20 cuestiones que sólo pueden ser contestadas con un Sí o un No.



Veinte preguntas-Newton.

Puedes pensar el proceso como un "árbol" que comienza en la pregunta inicial y se ramifica hacia abajo. Si Isaac Newton es el personaje famoso, la sucesión de respuestas podría empezar "sí, no, sí, sí,..." o, simplificando, "1011...".

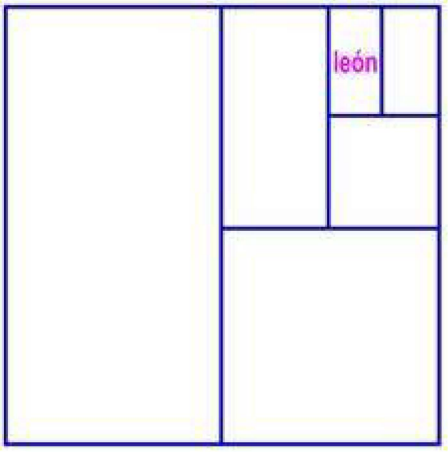
Hay 2 posibles respuestas para cada pregunta, por tanto $2^{20}=1.048.576$ posibles sucesiones. Asumiendo que el número de personajes famosos es menor que 10^6, el truco está en escoger las preguntas apropiadas para que dos de ellos no tengan las mismas sucesiones de ceros y unos. ¿Como deberíamos hacerlo?

Una mala primera pregunta sería: "¿Eres Tony Blair?". OK, podríamos ser afortunados y ganar a la primera, pero casi siempre la respuesta será No, y habríamos desperdiciado una pregunta porque no tendríamos mucha más información del personaje que al inicio del juego, habiendo descartado sólo una posibilidad.

Una buena primera pregunta sería: "¿Eres un hombre?". Cualquiera de las respuestas nos permitirá reducir las posibilidades a la mitad. Es como intentar capturar un león construyendo vallas que acoten a la mitad el lugar donde se encuentra: con independencia del lado del cercado en el que esté, cada vez nos quedamos con la mitad del espacio.

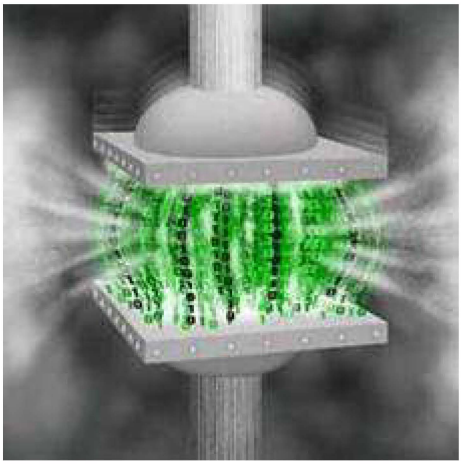
Escribiendo la totalidad del árbol (¡sobre una hoja de papel enorme!) tendríamos una forma de representar cada una de las $1.048.576$ personas como una sucesión de ceros y unos. Estos ceros y unos se denominan *bits*[1]. Se trata de una *codificación* de los personajes. (Cuando los teóricos de la información hablamos de codificación, no nos referimos a los códigos secretos como el Enigma alemán; de eso se ocupa un área diferente de las matemáticas, llamada *criptografía*).

Codificaciones eficientes



Capturando al león.

Una *codificación* es una regla que conecta una colección de objetos (como gente famosa o letras del alfabeto) con otra colección de cosas. Es como escribir una etiqueta para asignarla a cada objeto (generalmente usando sólo ceros y unos). La codificación más famosa es quizás el código Morse, que convierte letras del alfabeto en series de puntos y guiones.



Estamos interesados en codificaciones *eficientes*, esto es, en formas de convertir los mensajes en sucesiones cortas de ceros y unos. Suponemos que los mensajes fueron creados al azar, y queremos minimizar el promedio de bits necesarios. La forma en que lo hacemos es emparejar las letras del alfabeto cuya probabilidad de uso es mayor con sucesiones cortas de bits, y aquellas otras menos probables con sucesiones más largas. En Morse “E” es un sólo punto, mientras que “Q” es guión-guión-punto-guión.

Por ejemplo, en el juego de Veinte Preguntas, cada personaje famoso no tiene la misma probabilidad de ser el seleccionado; por tanto, necesitamos saber qué hacer si algunas posibilidades son más probables que otras. De hecho, ya sabemos qué hacer. Nos podemos atener al principio de que debemos dividir a la mitad en cada paso. El único cambio es que en lugar de intentar quedarnos en cada pregunta con la mitad de las personas, intentaremos reducir la probabilidad a la mitad cada vez. Esta es la idea que subyace en la *codificación de Huffman*.

Codificación de Huffman

En cada paso agrupamos los símbolos que tengan las probabilidades más bajas, y reordenamos la lista. Es más fácil entender esto con un ejemplo.

Supongamos que tenemos 5 letras: A aparece con probabilidad $1/2$, y B,C,D,E aparecen con probabilidad $1/8$ cada una.

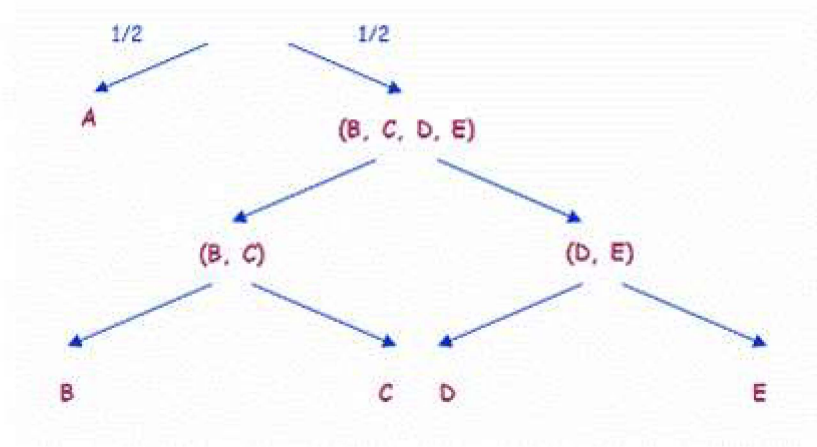
La codificación más simple sería escoger 5 sucesiones de 3 bits cada una y emparejarlas con las letras: digamos A=000, B=001, C=010, D=011, E=100. Entonces una letra media tendría 3 bits para describirla. Sin embargo, podemos hacerlo mucho mejor usando la codificación de Huffman.

Para crear la codificación de Huffman:

En el paso 1, agrupamos (D,E); entonces A tiene probabilidad $1/2$, (D,E) tiene probabilidad $1/4$ y B, C, tienen probabilidad $1/8$ cada una.

En el paso 2, agrupamos (B,C); entonces A tiene probabilidad $1/2$, (B,C) tiene probabilidad $1/4$ y (D,E) tiene probabilidad $1/4$.

En el paso 3, agrupamos $((D,E),(B,C))$; entonces A tiene probabilidad $1/2$, lo mismo que (B,C,D,E).



De nuevo podemos representar esto como un árbol y de nuevo podemos leer las etiquetas: A=1, B=011, C=101, D=001, E=000. Así, si enviamos una letra aleatoria, se tomará como promedio para describirla

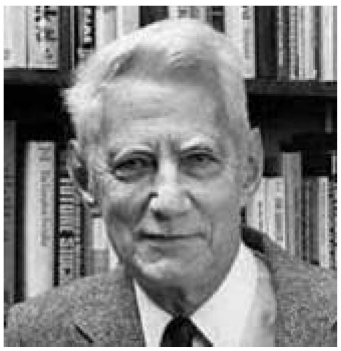
$$(1/2) \times 1 + (1/8) \times 3 + (1/8) \times 3 + (1/8) \times 3 + (1/8) \times 3 = 2$$

bits.

El proceso de reducción de 3 bits a 2 bits es lo que llamamos *compresión de datos*. Explotando nuestros conocimientos sobre probabilidades, podemos ahorrar espacio. Más aún, si codificamos más de una letra, este método da un mensaje *descifrable*; por ejemplo, si alguien almacena el mensaje 01101011000011 puede más tarde separarlo (empezando por la izquierda) en 011/010/1/1/000/011, ó *BCAAEB*.

Un hecho destacable es que esta simple codificación de Huffman ofrece el mejor rendimiento posible. Esto es, no existe ninguna codificación de letras individuales A, B, C, D, E en colecciones de ceros y unos que sea (i) descifrable y (ii) tenga una longitud promedio más corta con estas probabilidades. ¡Intenta comprobarlo! Además, esta propiedad es independiente de las probabilidades particulares consideradas. Para cualquier colección de letras y cualesquiera probabilidades, ¡la codificación de Huffman siempre será la mejor!

Entropía



Claude Shannon, padre de la teoría de la información.

Existe un límite para comprimir un globo relleno de aire, porque las moléculas no se pueden apretar tanto como uno quiera. Del mismo modo, existe un límite universal para comprimir datos. Esta cantidad de "relleno" del fichero es llamada *entropía*.

Shannon dio una fórmula general para calcular la entropía. Si las letras tienen probabilidades $p(1), p(2), p(3),..., p(r)$, necesitamos al menos

$$H = \sum_{s=1}^{r} [-p(s)\log_2 p(s)]$$

bits para expresar una letra (la expresión $\log_2$ denota un logaritmo en base 2). Esto sugiere que si una letra tiene probabilidad $p(s)$, deberíamos poder codificarla con una sucesión de, aproximadamente, $[-\log_2 p(s)]$ bits.

En el ejemplo anterior

$$H = -1/2 \times (-1) - 1/8 \times (-3) - 1/8 \times (-3) - 1/8 \times (-3) - 1/8 \times (-3) = 2,$$

así que la codificación de Huffman es realmente la mejor posible.

La entropía mide "cómo de aleatorio es algo". Al calcular la entropía vemos que el resultado de un 50% / 50% obtenido al lanzar una moneda no trucada (con entropía 1) es "más aleatorio" y, por tanto, más difícil de comprimir que el de una moneda trucada 90% / 10% (con entropía 0,46899).

Codificación universal

Parece que ya estuviera todo dicho. Sin embargo, al describir la codificación de Huffman hemos hecho una importante suposición, puesto que no siempre conocemos las probabilidades. Imaginemos, por ejemplo, que intentamos codificar un libro escrito en ruso donde ignoramos la probabilidad con la que aparece cada letra en cada etapa. O consideremos la primera foto de un extraño planeta: si todavía desconocemos a qué se parece, no podremos imaginar las probabilidades de que aparezca un color determinado.

Lempel y Ziv inventaron en la década de los 70 un método notable que evita completamente este problema. Recibe el nombre de procedimiento de compresión de datos *universal* porque es eficaz aun cuando no se conocen las probabilidades de las letras particulares. Funciona señalando sucesiones de letras que han aparecido antes, y usando las repeticiones para reconstruir el mensaje.

La regla es ir rompiendo el mensaje en palabras, eligiendo en cada etapa la palabra más corta que no hemos visto antes.

Si las letras son sólo A y B y el mensaje es

$$ABAABABAABAAB$$

colocamos una barra después de la primera letra, de modo que la primera palabra es A:

$$A/BAABABAABAAB$$

Ahora, el próximo símbolo B no es una palabra vista anteriormente, así que colocamos también una barra después de él. Hasta ahora hemos encontrado las palabras A y B:

$$A/B/AABABAABAAB$$

Como el símbolo A está ya en nuestra lista, la próxima posibilidad más corta es AA, de modo que ponemos una barra detrás. Continuando así, obtenemos eventualmente:

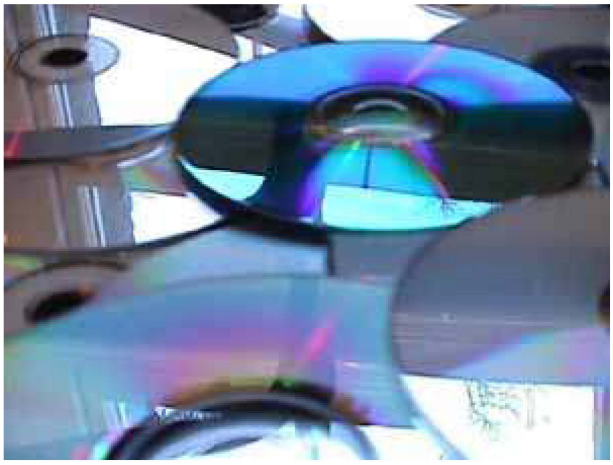
$$A/B/AA/BA/BAA/BAAB/$$

Ahora, al describir el mensaje, cada palabra será muy similar a una ya vista; por ejemplo, podemos escribir "$BAAB$" como "BAA" (una palabra ya vista) más "B" (una letra extra). Esto es, "Palabra $5 + B$".

Aunque pueda parecer poco eficiente, se demuestra que este procedimiento llega a serlo para mensajes suficientemente largos, incluso cuando no se conocen las probabilidades de las As y las Bs, y está próximo a la entropía.

Métodos modernos y el futuro

Muchos problemas de compresión de datos permanecen abiertos. Ideas como las codificaciones de Huffman y Lempel-Ziv están diseñadas para el caso *sin memoria*. Las tiradas de dados son sin memoria: ocurra lo que ocurra en el pasado, no afecta a las probabilidades de que en la próxima tirada salga un 6. Del mismo modo, la codificación de Huffman funciona cuando las letras se eligen aleatoriamente entre las letras ya escogidas, no afectando a lo que sucede la próxima vez.



Compresión de datos en acción.
[Imagen de DHD Photo Gallery].

Sin embargo, si queremos comprimir música o vídeos, no es este el caso. Por ejemplo, muy frecuentemente cuando un píxel es negro, los píxeles próximos a él serán negros (porque forman parte de una forma negra grande) y en el próximo cuadro el píxel también será negro (porque las cosas no cambian mucho de cuadro a cuadro). Mucha gente está trabajando con el fin de obtener la manera más eficiente de comprimir imágenes de vídeo, aprovechando propiedades como esta, para que los vídeos puedan ser enviados rápidamente a través de una línea telefónica.

Además, tenemos una cuestión de *pérdidas*. Los métodos anteriormente descritos producen una copia perfecta de los datos. Pero si admitimos una pequeña pérdida de calidad podemos obtener frecuentemente mejores resultados, como con los ficheros MP3.

A medida que la investigación continúa, los principios y el lenguaje diseñados por Shannon siguen estando en la base del trabajo.

---

[1] Los *bits* son los bloques constructores básicos de la moderna era de la información, así que vale la pena acostumbrarse a pensar sobre ellos. Aquí encontrarás una rápida explicación de lo que son los bytes, megabytes, etc., y algunas estimaciones de los requerimientos de almacenamiento de diversos tipos de información.

Sobre el autor

Oliver Johnson es miembro de Christ's College (como lo fueron Milton, Darwin, Richard Whiteley y Ali G). Llegó a Cambridge como estudiante en 1992 y desde entonces ha encontrado todo tipo de excusas para no marcharse. Investiga en teoría de probabilidades, tratando de entender la estructura que hay detrás de la aleatoriedad, para lo que se apoya en ideas relacionadas con este artículo.

matematicalia
revista digital de divulgación matemática

(*) Este artículo apareció en el número 23 (enero 2003) de *Plus Magazine*. *Matematicalia* agradece a los responsables del Millennium Mathematics Project de la Universidad de Cambridge la autorización para publicar su traducción al castellano. (Traductora: Edith Padrón).