Trabajo práctico 2
Codificación de Huffman

1. Introducción

1.1. Reseña histórica

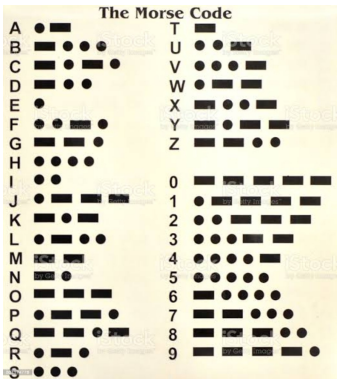

Figura 1: Código Morse.

Antes del desarrollo formal de la teoría de la información, la compresión de mensajes era un problema abordado empíricamente, sin una formulación matemática. Un ejemplo temprano es el *Código Morse* (1838), Figura 1, diseñado por Samuel Morse y Alfred Vail para el telégrafo eléctrico [1]. Este sistema asignaba secuencias cortas a letras frecuentes y más largas a letras poco comunes (para el idioma Inglés), anticipando el principio fundamental de lo que sería una codificación binaria eficiente: asignar menos bits (o señales) a los símbolos más probables.

Con la publicación del trabajo seminal de Claude Shannon [2], surgió una base teórica para la codificación de la información. Shannon demostró que la entropía $H(X)$ de una fuente X representa el límite inferior teórico para la longitud (en cantidad de bits) promedio de cualquier código sin pérdida de información. Poco después, Robert Fano propuso, junto con Shannon, el método de *Shannon–Fano coding* [3], que también construye un código que asigna una longitud variable a cada símbolo. Si bien este método mejora sustancialmente la eficiencia respecto de los códigos de longitud fija, no garantiza siempre la optimalidad global.

En 1951 David A. Huffman desarrolló el *código de Huffman* como parte de un trabajo práctico de la asignatura "teoría de la información" en el MIT para el profesor Robert M. Fano. La tarea consistía en encontrar el código binario más eficiente, y Huffman lo logró creando un método basado en un árbol binario ordenado por frecuencias. En 1952, Huffman publicó su artículo [4], en el cual presenta un algoritmo que construye un árbol binario de prefijos óptimo, bajo la restricción de que ningún código asignado sea prefijo[1] de otro. El trabajo de Huffman marcó un hito en la teoría de la información aplicada, estableciendo el estándar de codificación óptima sin pérdida. A partir de esta base, surgieron métodos más avanzados como la codificación aritmética [7] y los algoritmos de compresión adaptativos de Lempel y Ziv [5, 6], que extendieron las ideas de Huffman mediante modelado probabilístico dinámico y estructuras de diccionario. No obstante, la codificación de Huffman sigue siendo un componente esencial de numerosos estándares modernos como ZIP, PNG, JPEG, entre otros, tanto por su simplicidad como por su eficiencia teórica.

[1] El código de un símbolo se dice prefijo de otro si la cadena de bits que lo representa coincide con los primeros bits de la cadena que representa al otro símbolo. Ej: si $A \to 110$ y $B \to 11010$, entonces 110 es prefijo de 11010.

1.2. Idea básica de la códificación de Huffman

Un *código libre de prefijos* (*prefix-free*) es un tipo de codificación en el que ninguna palabra de código es prefijo de otra. Esto significa que ninguna secuencia binaria asignada a un símbolo puede coincidir con el inicio de la secuencia asignada a otro símbolo. Gracias a esta propiedad, la decodificación se puede realizar de manera *instantánea*, es decir, sin necesidad de un delimitador entre símbolos, ya que el receptor sabe exactamente cuándo termina cada palabra de código. El algoritmo de Huffman produce precisamente este tipo de códigos. Cada símbolo del alfabeto se asocia a una hoja en un árbol binario, como se muestra en la Figura 2. Para construir el código, se asigna el bit 0 a cada rama izquierda y el bit 1 a cada rama derecha (aunque esta convención puede invertirse sin afectar la validez del código). Este algoritmo de codificación se pueden resumir sintéticamente como en los siguientes pasos:

1. Se ordenan los símbolos por frecuencia, de menor a mayor.
2. Se toman dos símbolos (o subárboles) con menor frecuencia, se combinan en un nuevo nodo cuya frecuencia es la suma de ambas.
3. El proceso se repite iterativamente, combinando las hojas o subárboles de menor frecuencia para formar un nuevo subárbol.
4. Se concluye cuando queda formado un único árbol binario.
5. Finalmente, el código binario de cada símbolo se obtiene recorriendo el árbol desde la raíz hasta la hoja correspondiente, concatenando los bits de cada rama.

Ejemplo: suponer símbolos con sus frecuencias según la Tabla 1. En la Figura 2 se muestra como quedaría armado el árbol binario y en la tabla (columna central) el resultado de la codificación final asignada a cada símbolo.

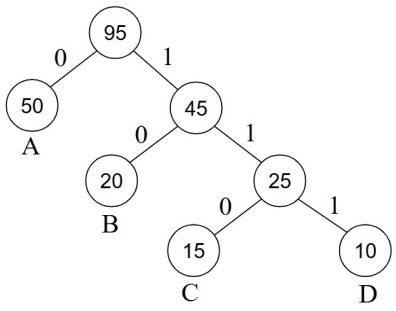

Figura 2

Símbolo	Código	Frecuencia
A	0	50
B	10	20
C	110	15
D	111	10

Tabla 1

Puede verificarse que ningún código es prefijo de otro, garantizando una codificación no ambigua. Esta estructura jerárquica es la base del **algoritmo de Huffman**, que asigna códigos más cortos a los símbolos más frecuentes y más largos a los menos frecuentes, minimizando así la longitud media del mensaje codificado.

2. Ejercicios

En este trabajo se aplicarán conceptos de teoría de la información al procesamiento de texto, utilizando el algoritmo de codificación de Huffman para analizar la eficiencia de compresión y la relación entre entropía, número promedio de bits y precisión en la estimación de probabilidades.

Ejercicio 1

Implemente una función en Python que reciba un texto y estime la masa de probabilidad $p(x)$ de aparición de cada carácter presente en el texto. La función debe retornar un diccionario cuya clave sea el caracter y su valor la probabilidad.

Genere un gráfico de barras mostrando la frecuencia o probabilidad estimada de cada carácter. Este análisis debe realizarse para tres textos distintos: `mitología.txt`, `tabla.txt` y `adn.txt`, provistos en el campus de la materia.

Ejercicio 2

Utilice la función `huffman_code()` provista por la cátedra para la codificación de Huffman, la cual recibe el diccionario de probabilidades (generado en el Ejercicio 1) y retorna un diccionario con el código de Huffman para cada caracter. Genere la secuencia de bits codificada para cada texto provisto, calcule y compare:

- La longitud promedio de bits (sin redondear) para la codificación no uniforme:

$$L = \sum_{i=1}^{N} p(x_i) l(x_i),$$

  donde N es la cantidad de símbolos diferentes en el texto, x_i el i-ésimo símbolo y $l(x_i)$ la longitud (cantidad de bits) asignada por el código de Huffman a ese símbolo.

- La cantidad de bits (enteros) mínima para representar los N símbolos del texto con una codificación uniforme (misma cantidad de bits para todos los símbolos presentes).

- El porcentaje de memoria reducido con Huffman respecto del la codificación uniforme.

Ejercicio 3

Calcule la entropía $H(X)$ de cada texto a partir de las probabilidades estimadas y compárela con la longitud promedio de codificación. Discuta qué tan cercanas son ambas magnitudes y qué representan. Extraiga conclusiones.

Ejercicio 4

Para comparar las entropías entre los distintos textos, debido a que cada uno posee una cantidad N de símbolos distinta, utilizaremos la entropía normalizada $\eta = H(X)/\log_2(N)$. Compare las entropías normalizadas entre los tres textos provistos y discuta los resultados teniendo presente las características de cada archivo y probabilidades de símbolos respectivos.

3. Conclusiones

Como conclusiones, elabore un resumen breve y conciso comentando características que considere relevantes del método propuesto en este trabajo y los resultados obtenidos, así como dificultades encontradas y cómo fueron abordadas.

4. Condiciones de entrega

- **Informe**: El informe debe entregarse en formato PDF (**no se aceptarán otros formatos**) y con nombre: `TP2_GXX.PDF` (donde `XX` es el número de grupo). Es condición necesaria para la aprobación cumplir con las pautas para la presentación de informes estipuladas en el campus.
- **Código**: El código debe incluirse junto al informe en un archivo ZIP (con mismo nombre que el informe) que deberá subirse al campus.
- Se recuerda a los estudiantes que las entregas deben ser un producto original de cada estudiante, por lo que se les pide revisar la sección 6 del programa de la materia y el Código de Honor y Ética de la Universidad.

Referencias

[1] A. Vail, *The Electro-Magnetic Telegraph: A Description of the Apparatus and the Processes of Telegraphing*, New York: D. Appleton & Co., 1858.

[2] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, pp. 379–423, 623–656, 1948.

[3] R. M. Fano, "The Transmission of Information," Technical Report No. 65, Research Laboratory of Electronics, Massachusetts Institute of Technology, 1949.

[4] D. A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[5] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 337–343, 1977.

[6] J. Ziv and A. Lempel, "Compression of Individual Sequences via Variable-Rate Coding," *IEEE Transactions on Information Theory*, vol. 24, no. 5, pp. 530–536, 1978.

[7] J. Rissanen and G. G. Langdon, "Arithmetic Coding," *IBM Journal of Research and Development*, vol. 23, no. 2, pp. 149–162, 1979.