TA137/86.25

Taller de Comunicaciones Digitales

Facultad de Ingeniería
Universidad de Buenos Aires

Simulación y Análisis de un Sistema de Comunicaciones

Trabajo Práctico N°1

Grupo N°3

2º Cuatrimestre 2025

Autores:	**Padrón:**	**Correo:**
Santiago Mogica	109026	smogica@fi.uba.ar
Sergio Aguirre	96953	saguirreg@fi.uba.ar
Tomás Giani	107629	tgiani@fi.uba.ar
Gonzalo Antahuara	109965	gantahuara@fi.uba.ar

Índice

1. Introducción **2**

2. Codificacion y decodificación de fuente **2**
2.1. Analisis Estadistico 2
2.2. Código Huffman 3
2.2.1. Codificación 3
2.2.2. Decodificación 3
2.2.3. Características del código obtenido 3
2.2.4. Comparación de resultados 4
2.3. Conclusiones 6

1. Introducción

El presente trabajo práctico tiene como objetivo simular y analizar un sistema de comunicaciones capaz de transmitir texto de un extremo a otro, siguiendo el esquema ilustrado en la **Figura 1.1**. En esta primera etapa, se implementó el algoritmo de Huffman con el propósito de generar un código binario que represente los caracteres del texto a transmitir. Para este propósito, se desarrolló un script en Python que permite ejecutar los procesos de codificación y decodificación sobre un archivo de texto.

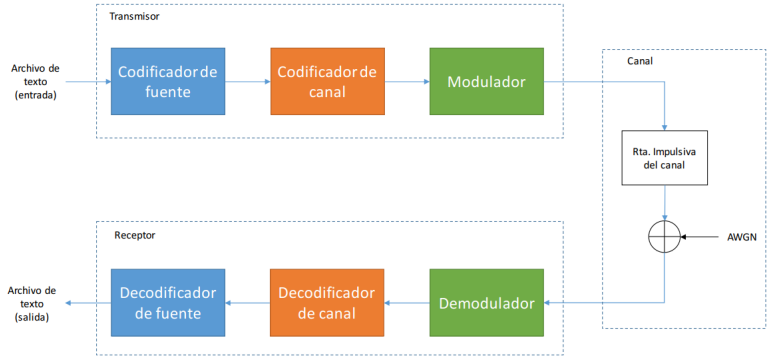

Figura 1.1: Diagrama de bloques del sistema de comunicación

2. Codificacion y decodificación de fuente

2.1. Analisis Estadistico

Para poder obtener el código de Huffman, primero calculamos la probabilidad de aparición de cada carácter en el archivo de texto. Para este análisis se utiliza como mensaje *"hola este es un ejemplo para usar Huffman"*. Se creo un código en donde se lee un archivo .txt y obtiene la frecuencia relativa de los caracteres de la siguiente forma:

$$p_{\text{carácter}} = \frac{\#\text{ Apariciones del carácter}}{\#\text{Caracteres totales}}$$

Habiendo obtenido todas las probabilidades de ocurrencia, se puede obtener la entropía de la fuente que este en nuestro caso es el texto ingresado. Para ello, se aplica la formula de la entropía de una fuente discreta y sin memoria ecuación

$$H(\mathbb{S}) = \sum_{k=0}^{K-1} P_k \log_2\left(\frac{1}{P_k}\right)$$

En donde $\mathbb{S}$ es el conjunto de caracteres presentes en el texto y P_k es la probabilidad asociada al k-ésimo carácter encontrado en el texto. Para el texto analizado la entropía es de: 3.9028 bits/símbolo.

2.2. Código Huffman

2.2.1. Codificación

A partir de las distribuciones de probabilidad obtenidas P_k, es posible construir un código casi óptimo mediante la aplicación del algoritmo de Huffman. Este algoritmo implementa una codificación de longitud variable para los caracteres, asignando códigos más cortos a aquellos con mayor probabilidad de ocurrencia.

2.2.2. Decodificación

Símbolo	**Probabilidad**	**Código**
a	0.1034	000
s	0.0517	0010
u	0.0517	0011
m	0.0517	0100
r	0.0517	0101
.	0.0172	01100
h	0.0345	01101
o	0.0690	0111
e	0.1207	100
l	0.0690	1010
t	0.0345	10110
n	0.0345	10111
p	0.0345	11000
f	0.0345	11001
j	0.0172	110100
g	0.0172	110101
i	0.0172	110110
d	0.0172	110111
␣	0.1724	111

Tabla 1: Símbolos, probabilidades y códigos asignados.

El proceso de decodificación es el inverso a la codificación, su objetivo es reconstruir el texto original a partir de la secuencia de bits generada. La decodificación es posible de manera unívoca y sin ambigüedad gracias a una propiedad fundamental del código Huffman: es un **código prefijo**. Esto significa que ninguna palabra de código es el prefijo de otra palabra de código más larga. El decodificador va leyendo una secuencia desde el primer bit, hasta que coincide con una palabra existente en la tabla. Una vez encontrado el símbolo, comienza a leer a partir del próximo bit nuevamente. Es relevante notar que el carácter con mayor probabilidad de ocurrencia, y debido a la forma de construcción de código Huffman, es el carácter espaciador.

2.2.3. Características del código obtenido

El algoritmo de Huffman genera un código eficaz para evitar pérdidas. En la Tabla 1 se muestra el código obtenido con sus probabilidades para cada símbolo. Sus características más importantes son:

- **Código de Longitud Variable:** A diferencia de códigos de longitud fija como ASCII (donde cada carácter ocupa, por ejemplo, 8 bits), el código Huffman asigna palabras de código de longitudes diferentes a cada símbolo de la fuente. Los símbolos más frecuentes tendrán las palabras de código más cortas, mientras que los símbolos menos frecuentes tendrán las más largas.
- **Código Prefijo:** Una propiedad del código Huffman es que ninguna palabra de código es prefijo de otra. Esto garantiza una decodificación instantánea y por ende, nuestro código será univocamente decodificable.
- **Eficacia:** Para una distribución de probabilidad de símbolos dada, el código Huffman es óptimo. No existe ningún otro código de símbolos con una longitud promedio de palabra de código menor que la obtenida por este algoritmo. Es la más cercana al límite (teórico) definido por la entropía de la fuente.
- **Pérdidas:** El proceso de codificación y decodificación con el algoritmo de Huffman es reversible, el texto original puede ser reconstruido a partir de la versión codificada, sin pérdida de información.

2.2.4. Comparación de resultados

A continuación, se realizará una comparación entre el código obtenido mediante el algoritmo de Huffman y un código de longitud fija, como el código ASCII. Para este análisis, se considerarán parámetros tales como la entropía de la fuente, la longitud promedio del código, la eficiencia del mismo y la longitud individual de cada código. La siguiente tabla muestra el código correspondiente a algunos de los símbolos ASCII de 8 bits.

Símbolo	**Código Binario**
A	01000001
B	01000010
C	01000011
a	01100001
b	01100010
c	01100011
0	00110000
1	00110001
2	00110010
@	01000000
#	00100011
&	00100110
(	00101000
)	00101001
+	00101011
-	00101101
=	00111101

Tabla 2: Símbolos ASCII y sus códigos en binario

Texto de entrada:

```
Oíd mortales el grito sagrado libertad libertad libertad oíd el ruido de rotas
↪ cadenas ved en trono a la noble igualdad se levanta a la faz de la tierra
↪ una nueva y gloriosa nación coronada su sien de laureles y a su planta
↪ rendido un león sean eternos los laureles que supimos conseguir coronados
↪ de gloria vivamos o juremos con gloria morir ved el trono a la noble
↪ igualdad ya su trono dignísimo abrieron las provincias unidas del sur y
↪ los libres del mundo responden al gran pueblo argentino salud al gran
↪ pueblo argentino salud y los libres del mundo responden al gran pueblo
↪ argentino salud
```

Texto decodificado:

```
Oíd mortales el grito sagrado libertad libertad libertad oíd el ruido de rotas
↪ cadenas ved en trono a la noble igualdad se levanta a la faz de la tierra
↪ una nueva y gloriosa nación coronada su sien de laureles y a su planta
↪ rendido un león sean eternos los laureles que supimos conseguir coronados
↪ de gloria vivamos o juremos con gloria morir ved el trono a la noble
↪ igualdad ya su trono dignísimo abrieron las provincias unidas del sur y
↪ los libres del mundo responden al gran pueblo argentino salud al gran
↪ pueblo argentino salud y los libres del mundo responden al gran pueblo
↪ argentino salud
```

Como era de esperar, al no agregar ruido, la codificación y decodificación se realizan de manera perfecta, sin errores.

Teniendo el alfabeto codificado, se pudo calcular tanto la longitud mínima como la longitud promedio del código. Cabe recordar que la longitud mínima se determina mediante la siguiente fórmula:

$$L_{\text{mín}} = \frac{H(\mathbb{S})}{\log_2 r}$$

Debido a que trabajamos con códigos binarios, r = 2, se puede definir la longitud mínima de la siguiente forma.

$$L_{\text{mín}} = \frac{H(\mathbb{S})}{\log_2 r} = H(\mathbb{S})$$

La entropía es la misma para ambos códigos, dado que la misma depende de la fuente, no del código, por lo cual, la $L_{\text{mín}}$ de ambas será igual.

La longitud media se calcula de la siguiente forma:

$$\bar{L} = \sum_{k=0}^{K-1} P_k L_K$$

En donde L_k es la longitud del código del k-ésimo carácter. Para el texto analizado se obtuvo $L_k = resultad$. Con estos parámetros se define la eficiencia como:

$$\eta = \frac{L_{\text{mín}}}{\bar{L}}$$

Parámetro	Huffman	ASCII
H $[\frac{bits}{simbolo}]$	3.9458	3.9458
$\bar{L}$ [bits]	3.9916	8
L_{min}[bits]	3.9458	3.9458
η	0.9885	0.4932

Tabla 3: Comparación de los parámetros de código.

2.3. Conclusiones

La implementación del algoritmo de Huffman permitió codificar un mensaje de manera eficiente, asignando códigos más cortos a los símbolos de mayor probabilidad, lo cual redujo notablemente la longitud promedio del mensaje transmitido en comparación con un esquema de codificación de longitud fija como ASCII.

En los resultados obtenidos se observó que la longitud promedio de codificación con Huffman fue de $\bar{L} = 3{,}9916$ bits por símbolo, en contraste con los 8 bits por símbolo requeridos por ASCII. Esta diferencia representa una mejora significativa en términos de compresión y uso del canal. Además, la eficiencia del código de Huffman fue de $\eta = 0{,}9885$, muy cercana al valor ideal, y considerablemente superior a la eficiencia obtenida con ASCII ($\eta = 0{,}4932$).

En este caso, al ser una fuente con relativamente pocos caracteres, el rendimiento de la codificación ASCII es bajo debido a que se utilizan, en proporción, pocos símbolos a comparación del total de símbolos que esta codificación incluye.

Finalmente, se verificó que el proceso de decodificación reconstruye el mensaje original de forma exacta, gracias a que el código de Huffman es un código prefijo. Esto garantiza la decodificación sin ambigüedades y valida la correcta implementación del sistema de comunicación simulado, sumado a que no hay adición de ruido externo.

En conclusión, el trabajo permitió demostrar la efectividad de los códigos de longitud variable y el impacto positivo del análisis estadístico de la fuente en la optimización de los sistemas de codificación.