Taller 3 Comunicaciones Digitales

Elías Courdin
Comunicaciones Digitales 2025
Facultad de Ingeniería
Montevideo, Uruguay
eliascourdin@15gmail.com

Uriel Yaffé
Comunicaciones Digitales 2025
Facultad de Ingenieria
Montevideo, Uruguay
urielyaffe@gmail.com

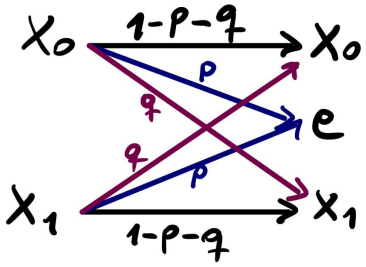

Fig. 1. Diagrama de canal binario simétrico con borrado

I. Canal simétrico con borrado

En un canal binario simétrico con borrado la fuente X emite símbolos de un alfabeto $A = \{x_0, x_1\}$. En el canal se introduce ruido, y el receptor decide entre tres símbolos posibles a la salida.
La variable aleatoria Y en el receptor toma valores en el alfabeto $B = \{x_0, e, x_1\}$. El símbolo e indica "$borradura$" y corresponde a una zona de detección entre x_0 y x_1 donde el receptor no se arriesga por ninguno de los dos. Los símbolos a la entrada del canal se generan en forma independiente y con las siguientes probabilidades:

$$P(X = x_1) = a$$
$$P(X = x_0) = 1 - a$$

Las probabilidades condicionales con que se reciben los símbolos son:

$$P(Y = x_0 \mid X = x_0) = P(Y = x_1 \mid X = x_1) = 1 - p - q$$
$$P(Y = x_1 \mid X = x_0) = P(Y = x_0 \mid X = x_1) = q$$
$$P(Y = e \mid X = x_0) = P(Y = e \mid X = x_1) = p$$

Un diagrama ilustrativo del canal se observa en la figura 1

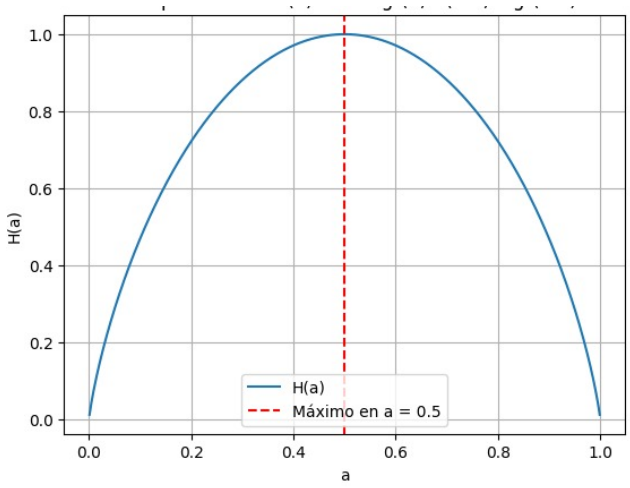

Fig. 2. Gráfica de la entropía de la fuente en función de a

A.

Para hallar la probabilidad de que un símbolo sea borrado por el algoritmo de detección planteamos:

$$P(Y = e|X) = \sum_x P(x)P(Y = e|X = x)$$
$$= aP(Y = e|X = x_0) + (1 - a)P(Y|X = x_1)$$
$$\Rightarrow P(Y = e|X) = ap + (1 - a)p = p$$

B.

Para calcular los bits de información por símbolo transmitido planteamos la expresión para la entropía de la fuente:

$$H(X) = -\sum_{x \epsilon \mathcal{X}} P(x) log_2(P(x)) = -a log_2(a) - (1-a) log_2(1-a)$$
$$\Rightarrow H(X) = \Omega(a)$$

C.

En la figura 2 Se observa la gráfica de la entropía de la fuente en función de a. Se puede ver como la entropía se maximiza para $a = \frac{1}{2}$, lo que indica que es el valor de a que genera mayor incertidumbre dado que los símbolos a la entrada son equiprobables. Para este caso, $H(\frac{1}{2}) = 1$ bit de información.

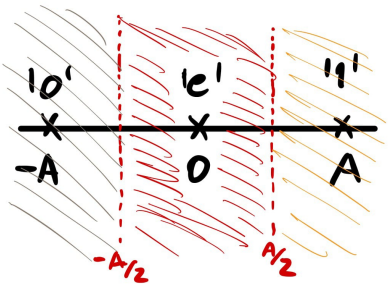

Fig. 3. Constelación BPSK modificada en recepción

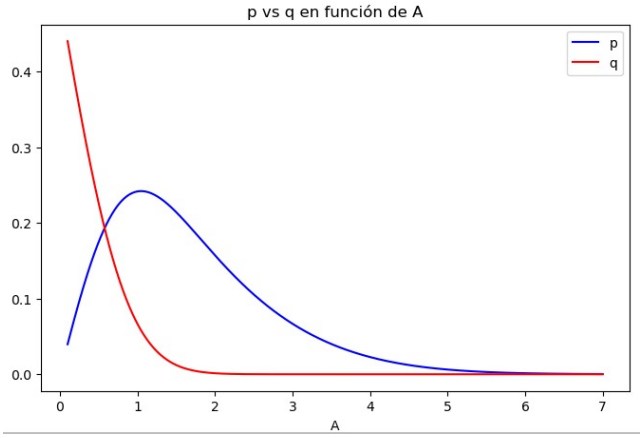

Fig. 4. p vs q en función de A ($\sigma = 1$)

D.

Se podría modificar la constelación en recepción para un sistema BPSK como se observa en la figura 3. Se agrega el símbolo e en el punto medio entre $-A$ y A, es decir, el punto de mayor incertidumbre a la hora de decidir. En esta imagen también se observan las regiones de decisión. Aunque la región de decisión es un tanto exagerada, ya que se podría haber tomado un pequeño entorno alrededor de "e", sin embargo, creemos que es una elección que simplifica el análisis del problema.

Considerando que el ruido introducido en el canal es blanco y gaussiano (AWGN), calculamos las probabilidades p y q en función de parámetros del sistema.

Sin pérdida de generalidad, haremos los cálculos suponiendo que se envía $-A$ (región izquierda en la figura 3). Podemos plantear lo siguiente para la probabilidad p:

$$p = P\left(\frac{A}{2} < N_D < \frac{3A}{2}\right)$$

Con $N_D \sim \mathcal{N}(0, \sigma^2)$

$$\Rightarrow p = Q\left(\frac{\frac{A}{2}}{\sigma}\right) - Q\left(\frac{\frac{3A}{2}}{\sigma}\right)$$

Siguiendo un razonamiento similar, hallamos la probabilidad q como:

$$q = P\left(\frac{3A}{2} < N_D\right) = Q\left(\frac{\frac{3A}{2}}{\sigma}\right)$$

E.

A partir de ahora nos enfocaremos en una situación de SNR razonable, por lo que p será pequeño.

Que p sea pequeño, significa que la probabilidad de que el ruido blanco gaussiano de media cero y varianza σ^2 esté entre $\frac{A}{2}$ y $\frac{3A}{2}$ es pequeña. Suponiendo que A es lo suficientemente grande, entonces q es aún más pequeña que p, y por lo tanto despreciable.

En la figura 4 se observa la gráfica de p vs q en función de A, suponiendo $\sigma = 1$. Se puede ver como para valores pequeños de A, q es más grande que p, pero a medida que A crece esto se revierte. Si A es lo suficientemente grande, ambos son despreciables.

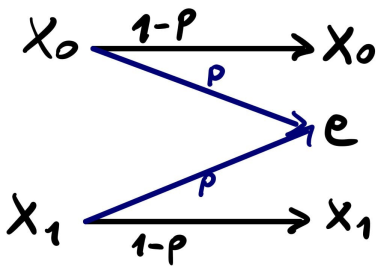

Fig. 5. Canal con borradura

Al asumir que q es despreciable, el nuevo modelo de canal asume que en recepción solo puede haber acierto o borrado, ya no hay error. El diagrama para este canal se observa en la figura 5 y se conoce como canal con borradura.

F.

Dado este nuevo modelo de canal, se calculan los siguientes valores en función de los parámetros del sistema.

$$H(Y) = \sum_{y \in \{X_0, e, X_1\}} -P(Y = y) log_2\left(P(Y = y)\right)$$

$$H(Y) = -a(1-p) log_2\left[a(1-p)\right]$$

$$-(1-a)(1-p) log_2\left[(1-a)(1-p)\right] - (p) log_2\left(p\right)$$

Desarrollando y agrupando términos, se llega a:

$$H(Y) = \Omega(p) - (1-p)\Omega(a)$$

con

$$\Omega(p) := -p log_2(p) - (1-p) log_2(1-p)$$

La idea intuitiva de $H(Y)$ es la incertidumbre que tiene la salida del canal, con cuantos bits se puede representar la salida.

La entropía de la salida dada la entrada es:

$$H(Y|X) = \sum_{x \in \{X_0, X_1\}} P(X = x) H(Y|X = x)$$

$$\Rightarrow H(Y|X) = a\Omega(p) + (1-a)\Omega(p) = \Omega(p)$$

La idea intuitiva de $H(Y|X)$ es la incertidumbre que se tiene en la salida ya sabiendo cuál fue la entrada.

La diferencia entre las dos entropías anteriores es que al condicionar la salida con la entrada, esta entropía es igual o menor a la de la salida, ya que al aportar la información de X, este puede reducir o no afectar la incertidumbre de la salida.

La entropía de la entrada dada la salida es:

$$H(X|Y) = \sum_{y \in \{X_0, e, X_1\}} P(Y = y) H(X|Y = y)$$

Observando la figura 5 podemos deducir que solo hay incertidumbre sobre la entrada viendo la salida cuando se recibe "e".

Entonces:

$$H(X|Y) = P(Y = e) H(X|Y = e)$$

$$= [ap + (1-a)p]\, \Omega(a) = p\Omega(a)$$

La idea intuitiva de $H(X|Y)$ es que mide la incertidumbre que queda sobre la entrada X una vez que se conoce la salida Y. En este caso particular, como se muestra en la Figura 5, si se recibe x_0 o x_1, no hay incertidumbre: sabemos exactamente cuál fue el símbolo transmitido, ya que no hay errores de transmisión (solo hay posibilidad de borrado). Sin embargo, si se recibe un símbolo de borradura, sí hay incertidumbre: no sabemos si se borró un x_0 o un x_1. Esa es la única fuente de entropía condicional en este canal.

Sabiendo $H(X)$ y $H(X|Y)$, podemos hallar la información mutua:

$$I(X;Y) = H(X) - H(X|Y) = \Omega(a) - p\Omega(a)$$

$$\Rightarrow I(X;Y) = (1-p)\Omega(a)$$

Con esto podemos hallar la capacidad del canal, que se obtiene maximizando la mutua:

$$C = max_{p(x)} I(X;Y) = 1 - p$$

El máximo se alcanza cuando $\Omega(a) = 1$, esto es, si $a = \frac{1}{2}$. Cuando la incertidumbre en la entrada es máxima, el canal alcanza su capacidad.

La información mutua $I(X;Y)$ representa cuánta incertidumbre se reduce sobre una variable al observar la otra, o dicho de otra forma, cuánta información aporta la salida sobre la entrada o viceversa. En este canal, como no hay errores y solo existe la posibilidad de borrado, la información mutua se ve reducida únicamente por los símbolos que se borran. Cuando no hay borraduras ($p = 0$), toda la información se transmite sin pérdida ($I(X;Y) = \Omega(a) = 1$). Cuando hay más borraduras, se pierde más información.

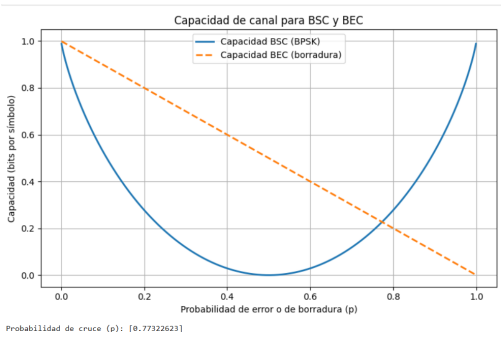

Fig. 6. Gráfica de comparación de capacidades

G.

Un canal BPSK "tradicional", se compone de dos posibles salidas en las cuales la entrada puede valer una o la otra. Este canal es un canal binario y simétrico (BSC), que tiene únicamente probabilidad de error (un símbolo que fue decodificado mal).

El canal BPSK al no contar con la posibilidad de borrado, tiene una capacidad de:

$$C = max_{p(x)} I(X;Y) = \Omega(a)(1 - \Omega(q))$$

con q la probabilidad de error.
Maximizando C, con a=$\frac{1}{2}$, $C = 1 - \Omega(p)$.

La capacidad para el sistema anterior se encuentra hallando la densidad de probabilidad $p(x)$ que maximiza la $I(X,Y)$. Por lo hallado en la parte I-F, tenemos que:

$$C = max_{p(x)} I(X;Y) = (1-p)\Omega(a)$$

. Maximizando C, con a = $\frac{1}{2}$, se obtiene:

$$C = 1 - p$$

Si igualamos la capacidad de ambos canales, con $p = q$ (donde p es la probabilidad de error en el canal BSC y q la de borrado en el canal BEC). Así, la comparación entre sus capacidades en función de p se muestra en la Figura 6.

El cruce entre ambas capacidades se da en $p = 0.773$, lo que significa que con una probabilidad de borrado menor a 0,773 (Con $P_{Borrado} = P_{error}$), el sistema con borradura llega a una mejor capacidad que el BPSK.

II. Algoritmos de decisión iterativa

Sea X el símbolo aleatorio que representa la palabra oculta. Sean x_k la palabra que ingresamos en el intento k-ésimo, Y_k el patrón (aleatorio) correspondiente y X_k el conjunto de palabras que concuerdan con los patrones $y_1, y_2, \ldots, y_k$ que hemos observado hasta ahora.

A.

Y_k es el patrón aleatorio correspondiente luego de introducir como intento la palabra x_k. Aunque parezca Y_k una variable aleatoria, es una función determinística de X y x_k, con X la palabra secreta. Si sabemos cuál es la palabra secreta y qué entrada se introdujo, entonces se puede calcular los colores del patrón de Y_k. Si la letra en la posición i de x_k coincide con la letra de la posición i de X, entonces y_{k_i} va a ser verde, si la letra de la posición i de x_k no coincide con la letra en la posición i de X, pero esa letra existe en alguna posición de X, entonces y_{k_i} va a ser amarillo y si no pasa ninguna de las dos cosas anteriores, y_{k_i} será roja.

B.

$H\{X|Y_k\}$ Es la incertidumbre de la palabra secreta dado que sabemos el color del patrón Y_k luego del intento x_k. Por ejemplo, si Y_k es [Rojo,Rojo,Rojo], sabemos que ninguna de las tres letras que se colocaron forman parte de la palabra secreta. Si asumimos que se colocaron tres letras diferentes (En realidad no tienen que ser todas diferentes, sino que lo hacemos para simplificar las cuentas y que el ejemplo sea más entendible), entonces de las 26 letras del alfabeto , nos quedan 23 para poder intentar, asumiendo la equiprobabilidad de la palabra secreta P($x_{k+1} = X$) = $\frac{1}{23*22*21}$. Pero si en la salida de Y_k es [Verde,Verde,Verde], entonces ya se averiguó cuál es la palabra secreta y la $H\{X|Y_k\} = 0$

C.

Como la palabra secreta se encuentra uniformemente distribuida en X_{k-1} (EL conjunto de entradas posibles luego de observar y_{k-1}), entonces, $P(X = x|Y_k = y_k) = \frac{1}{|X_k|}$ para x$\in X_k$. Lo que implica que:

$$H\{X|Y = y_k\} = log_2(|X_k|)$$

Por definición de entropía condicional:

$$H\{X|Y_k\} = \sum_{y_k \in Y} P(Y = y_k).H\{X|Y = y_k\}$$

.

Sustituyendo las últimas dos expresiones:

$$H\{X|Y_k\} = \sum_{y_k \in Y} P(Y = y_k).log_2(|X_k|)$$

Como $\sum_{y_k \in Y} P(Y = y_k).X$ se interpreta como la esperanza de X sobre la variable aleatoria y_k, ya que se puede ver como la sumatoria de y_k posibles por su probabilidad.

$$H\{X|Y_k\} = E_{Y_k}[log_2(|X_k|)]$$

D.

Nuestra elección de los x_k deberá maximizar la entropía de Y_k, esto se debe a que se busca que la información que nos brinda haber visto Y_k nos reduzca la cantidad de opciones de palabras lo mayor posible. Desarrollando la información mutua entre X , Y_k:

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

Juntando las dos últimas igualdades y despejando,

$$H(X|Y) = H(X) + H(Y|X) - H(Y)$$

Como se habló en la parte (a), $Y_k = f(X, x_k)$, por lo tanto, si se conoce la palabra secreta, y la entrada (x_k), se deduce el patrón de colores Y_k , por lo tanto $H(Y|X) = 0$.
Entonces para minimizar $H(X|Y)$, tenemos que maximizar $H(Y)$, maximizando así $I(X;Y)$.

E.

La probabilidad que tiene cada patrón Y_k depende de la cantidad de palabras que lo generan. Por ejemplo, el patrón [Verde, Verde, ... ,Verde] es generado por una única palabra, la palabra correcta. Sin embargo, hay otros patrones que son generados por más de una palabra.
La probabilidad de obtener un patrón y_k dado el intento x_k puede expresarse como sigue:

$$P(Y_k = y_k|X_k = x_k) = P(y_k = f(X, x_k)$$

Como X está uniformemente distribuida en X_k, y considerando que hay varias palabras que pueden generar un patrón, tenemos:

$$P(Y_k = y_k|X_k = x_k) = \frac{\#\ x_k \in X_k \text{ que generan } y_k}{|X_k|}$$

Donde la aleatoriedad está en la palabra oculta $X \in X_k$.

Dado esto, se puede calcular la entropía de Y_k para un intento x_k como:

$$H(Y_k \mid x_k) = -\sum_{y_k} P(Y_k = y_k \mid x_k) \cdot \log_2 P(Y_k = y_k \mid x_k)$$

donde la suma recorre todos los patrones posibles que pueden surgir al comparar x_k con las palabras en X_k.

F.

El script simula múltiples partidas de Wordle, comenzando siempre con una misma palabra inicial seleccionada previamente. A partir del patrón que se genera tras cada intento, el algoritmo tiene dos opciones para continuar: puede elegir aleatoriamente una palabra del conjunto de candidatas compatibles con el patrón Y_k, o bien seleccionar la que maximice la entropía de Y_k.
En la Figura 7 se observa que la estrategia basada en la maximización de la entropía obtiene mejores resultados en promedio (Primer intento sin maximizar entropía y el segundo maximizando). Todas las simulaciones comienzan con la misma palabra inicial, ya que algunas palabras contienen letras más frecuentes que otras y así nos aseguramos que empiezan ambas estrategias en las mismas condiciones.
Luego de varios intentos, notamos que la diferencia de intentos promedios entre maximizando la entropía y eligiendo al azar una palabra es de 0,7. Se necesitan 0,7 intentos menos si maximizamos la entropía.

Al seleccionar palabras al azar entre las disponibles, se necesitan en promedio entre 4,5 y 5 intentos para encontrar

Fig. 7. Simulación de varios intentos de Wordle

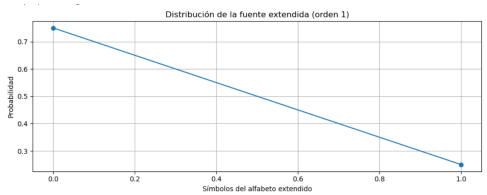

Fig. 8. Distribución de la fuente extendida con orden 1 para mensaje1.txt

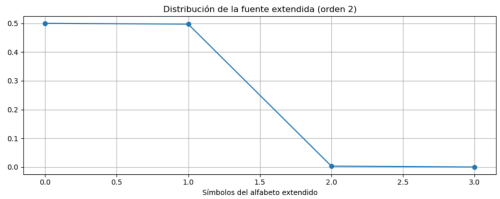

Fig. 9. Distribución de la fuente extendida con orden 2 para mensaje1.txt

la solución, dependiendo de la palabra inicial elegida. Dado que una partida de Wordle permite hasta 6 intentos, este desempeño es comparable al de un jugador humano promedio.

III. Compresión de la fuente

En esta parte se trabajará con extensiones de la fuente y el algoritmo de Huffman para verificar su capacidad de compresión, haciendo uso de archivos de texto y scripts de Python.

A.

La palabra 'óptimo' está en comillas porque el código Huffman te asegura encontrar un prefijo de código cuya longitud en promedio es más corto o igual que cualquier otro código. Esto se cumple bajo ciertas hipótesis, entre ellas que los símbolos estén idénticamente distribuidos.

En textos reales, esta hipótesis no se cumple del todo. Aunque las letras tienen distintas probabilidades de aparición ,no son independientes. Por ejemplo, en el español, después de una 'q' es casi seguro que venga una 'u'. Es decir, hay una dependencia entre los símbolos que Huffman no tiene en cuenta.

B.

Entropías	**adams**	**mensaje1**	**mensaje2**
Fuente ext. 1 vez	0.9937269	0.8112781	1.0
Huffman ext. 1 vez	0.9937269	0.8112781	1.0
Fuente ext. 2 veces	0.9782744	0.5132066	0.9999541
Huffman ext. 2 veces	0.9791655	0.9196101	0.9999662
Fuente ext. 4 veces	0.8425417	0.2632066	0.9994007
Huffman ext. 4 veces	0.9952297	0.9196103	0.9995912
Fuente ext. 8 veces	0.5539221	0.0066033	0.9897617
Huffman ext. 8 veces	0.9952713	0.0525637	0.9986161

TABLE I
Comparación de la entropía por bit y codificación Huffman para diferentes extensiones de la fuente

El código toma un mensaje y lo traduce a binario utilizando la codificación ASCII. Luego agrupa los bits según el orden de extensión y calcula la distribución de probabilidad de cada una de esas agrupaciones de bits.

Para comprender mejor las gráficas, analizaremos el archivo mensaje1.txt, que contiene únicamente el símbolo 'A'. El código ASCII de la letra A es 65, cuya representación binaria es 01000001.

Cuando se trabaja con una fuente extendida de orden 1, se consideran los bits individuales como símbolos, es decir, el alfabeto es $0, 1$. En la palabra 01000001 hay seis ceros y dos unos, por lo tanto, en la Figura 8 se observa que el símbolo 0 tiene una probabilidad de 0,75 y el símbolo 1 una probabilidad de 0,25.

Al extender la fuente a orden 2, se agrupan los bits de a dos, formando símbolos del alfabeto $00, 01, 10, 11$. En este caso, al recorrer la palabra 01000001 de a pares de bits, aparecen dos veces el símbolo 00 y dos veces el símbolo 01, por lo que en la Figura 9 se representan los símbolos correspondientes (en decimal) con probabilidades iguales de 0,5.

Siguiendo esta lógica, para la extensión de orden 4 se agrupan los bits de a cuatro. En el caso de 01000001, se forman dos bloques: 0100 y 0001, cada uno con una ocurrencia. Por lo tanto, en la Figura 10 aparecen dos símbolos con probabilidad 0,5 cada uno.

Finalmente, al extender la fuente a orden 8, se agrupan los bits de a ocho. Como todo el archivo contiene únicamente letras A, solo aparece una vez la secuencia 01000001, y se repite a lo largo del mensaje. Por esta razón, en la Figura 11 se observa un único símbolo con probabilidad 1, mientras que los demás tienen probabilidad cero. El símbolo representado tiene un valor de 65, lo que confirma que se trata del código ASCII de la letra A.

C.

Al analizar las longitudes de los códigos Huffman generados, se puede notar que, en los textos donde los símbolos

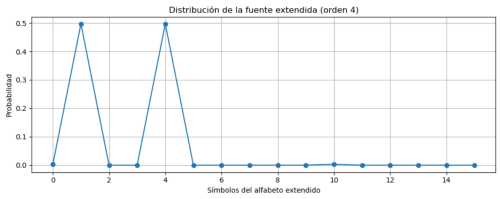

Fig. 10. Distribución de la fuente extendida con orden 4 para mensaje1.txt

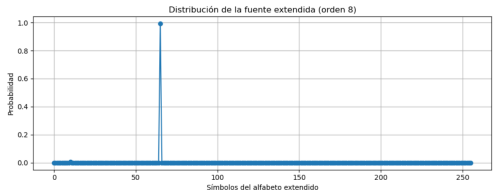

Fig. 11. Distribución de la fuente extendida con orden 8 para mensaje1.txt

están más equilibradamente distribuidos, las longitudes de los códigos resultantes tienden a mantenerse dentro de un rango acotado. Es decir, no hay símbolos que reciban códigos extremadamente cortos ni otros que tengan códigos muy largos.

En cambio, en el caso de mensaje1, que contiene únicamente la letra A, se observa un comportamiento muy distinto. Para la fuente extendida de orden 8, se asigna el código 1 al símbolo 01000001 (65 en decimal), correspondiente a la letra A, que es el único que aparece en el texto. Como los otros 255 posibles símbolos no se encuentran en el mensaje, el algoritmo de Huffman les asigna códigos arbitrarios, generalmente extendiendo el código más corto al agregarle más bits. Lo que genera que el único símbolo presente tiene un código de solo un bit, mientras que los símbolos ausentes terminan con códigos que pueden alcanzar hasta 255 bits de longitud. Esta situación se observa en la Figura 12.

Mientras que para el mensaje2.txt, donde los distintos símbolos aparecen con frecuencias similares, las longitudes de los códigos Huffman no varían tanto. En la fuente extendida de orden 8, como todos los símbolos tienen probabilidades parecidas, no hay uno que destaque significativamente sobre los demás. Por eso, no es necesario asignar un código especialmente corto a un único símbolo, y las longitudes de los códigos Huffman resultan más equilibradas, oscilando entre 7 y 10 bits.

```
Código de Huffman ordenado por longitud del código:
Símbolo: 01000001, Código: 1, Longitud: 1
Símbolo: 00001010, Código: 01, Longitud: 2
Símbolo: 00000000, Código: 001, Longitud: 3
Símbolo: 00000001, Código: 0001, Longitud: 4
Símbolo: 00000010, Código: 00001, Longitud: 5
Símbolo: 00000011, Código: 000001, Longitud: 6
Símbolo: 00000100, Código: 0000001, Longitud: 7
Símbolo: 00000101, Código: 00000001, Longitud: 8
Símbolo: 00000110, Código: 000000001, Longitud: 9
Símbolo: 00000111, Código: 0000000001, Longitud: 10
Símbolo: 00001000, Código: 00000000001, Longitud: 11
Símbolo: 00001001, Código: 000000000001, Longitud: 12
Símbolo: 00001011, Código: 0000000000001, Longitud: 13
Símbolo: 00001100, Código: 00000000000001, Longitud: 14
Símbolo: 00001101, Código: 000000000000001, Longitud: 15
Símbolo: 00001110, Código: 0000000000000001, Longitud: 16
Símbolo: 00001111, Código: 00000000000000001, Longitud: 17
```

Fig. 12. Código huffman para mensaje1 con orden 4

Por último, en el caso de adams.txt, al tratarse de un texto en inglés, cada letra tiene una determinada probabilidad de aparición. No se trata de un caso extremo como mensaje1, donde aparece un solo símbolo, ni tampoco como mensaje2, donde los símbolos están distribuidos de manera casi uniforme.

En este caso, algunas letras y símbolos aparecen con mayor frecuencia que otros. El símbolo más frecuente, según la fuente extendida de orden 8, es el 00100000 (32 en decimal), que corresponde al espacio. Este carácter se repite muchas veces a lo largo del texto.

El código Huffman generado refleja estas frecuencias: por ejemplo, hay nueve palabras de código con longitud 4, cuatro de longitud 5, ocho de longitud 6, y así sucesivamente hasta llegar a palabras de longitud 12. A partir de esa longitud, como ya no quedan símbolos con probabilidad positiva, el algoritmo asigna códigos arbitrarios a los símbolos restantes, cada vez más largos, llegando hasta una palabra de código de longitud 217. Esto se debe a que, al no haber ocurrencias de esos símbolos en el texto, se les asigna un código solo por completitud del árbol, sin que representen información real.

D.

Se presenta una tabla comparando el tamaño de los archivos comprimidos mediante el algoritmo de Huffman (en bits y bytes) y mediante ZIP (en bytes):

Archivo	**Huffman (bits)**	**Huffman (bytes)**	**ZIP (bytes)**
ADAMS	8428	1053.5	1216
Mensaje1	168	21	209
Mensaje2	16281	2035.125	2255

TABLE II
COMPARACIÓN DEL TAMAÑO DE ARCHIVOS COMPRIMIDOS CON HUFFMAN Y ZIP.

El algoritmo de Huffman obtiene una mejor compresión que ZIP en los tres casos si se considera solo el tamaño codificado.

Esta comparación no es completamente justa, ZIP incorpora metadatos, cabeceras y otras técnicas de compresión, lo que explica su tamaño más grande, pero también su mayor generalidad y robustez.

IV. CODIFICACIÓN PARA CORRECCIÓN DE ERRORES

En esta sección se realizan los ejercicios 9.4, 9.6 y 9.7 del capítulo 9 de las notas de Belzarena y Larroca[1].

A. Ejercicio 9.4

- Sea C un código lineal definido por la siguiente matriz de paridad:

$$\mathbb{H} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Si se recibe la palabra 110110 y solo se comete un error ¿cuál es la palabra de código enviada?.

$*$ Para ver cuál es la palabra de código enviada, si sabemos que se cometió un único error, comparamos las palabras de las filas de H, con la palabra recibida.
Para esto sumamos esas palabras con la recibida y como la suma es módulo 2, el resultado será un vector con 1 en las posiciones que hubo un error. Nos quedamos con la palabra que haya dado error en un solo bit. La palabra es (1,1,0,0,1,0), la palabra recibida difiere solamente en el cuarto bit.

• Ahora se considera un código (7, 4) con matriz generadora:

$$\mathbb{G} = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

1) Encuentre todas las palabras de código.
$*$ Como tenemos un código lineal definido por su matriz generadora, las palabras de código se obtienen de hacer las combinaciones lineales entre los vectores generadores que conforman las filas de $\mathbb{G}$. Se obtiene la siguiente lista:

$$\begin{Bmatrix} (0 & 0 & 0 & 0 & 0 & 0 & 0) \\ (0 & 1 & 0 & 1 & 1 & 0 & 0) \\ (1 & 0 & 1 & 0 & 1 & 0 & 0) \\ (0 & 1 & 1 & 0 & 0 & 1 & 0) \\ (1 & 1 & 0 & 0 & 0 & 0 & 1) \\ (1 & 1 & 1 & 1 & 0 & 0 & 0) \\ (0 & 0 & 1 & 1 & 1 & 1 & 0) \\ (1 & 0 & 0 & 1 & 1 & 0 & 1) \\ (1 & 1 & 0 & 0 & 1 & 1 & 0) \\ (0 & 1 & 1 & 0 & 1 & 0 & 1) \\ (1 & 0 & 1 & 0 & 0 & 1 & 1) \\ (1 & 0 & 0 & 1 & 0 & 1 & 0) \\ (0 & 0 & 1 & 1 & 0 & 0 & 1) \\ (0 & 1 & 0 & 1 & 0 & 1 & 1) \\ (0 & 0 & 0 & 0 & 1 & 1 & 1) \\ (1 & 1 & 1 & 1 & 1 & 1 & 1) \end{Bmatrix}$$

2) ¿Cuál es la distancia mínima del código?
$*$ La distancia mínima del código corresponde al peso de la palabra con menor peso del código, siendo el peso de la palabra igual a la cantidad de ”unos” que tiene el vector. Observando el conjunto, vemos que la distancia mínima para este código es 3.
3) Encuentre la matriz de paridad $\mathbb{H}$ del código.

$*$ La matriz de paridad puede construirse a partir de la matriz generadora $\mathbb{G}$. Si escribimos a $\mathbb{G}$ de la forma sistemática como:

$$\mathbb{G} = [\mathbb{I}_k | \mathbb{P}]$$

entonces se obtiene la matriz de paridad traspuesta:

$$\mathbb{H}^T = \begin{bmatrix} \mathbb{P} \\ \mathbb{I}_{n-k} \end{bmatrix}$$

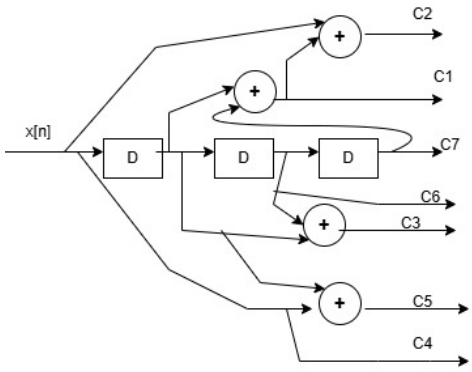

Fig. 13. Diagrama de bloques para el código (7,4)

.
Teniendo en cuenta que se pueden intercambiar filas y columnas de $\mathbb{G}$, además de realizar combinaciones lineales de sus filas para llevarla a su forma sistemática sin alterarse su propiedad de matriz generadora. Se obtiene:

$$\mathbb{H} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Haciendo $fila_1 = fila_1 + fila_2$ en $\mathbb{G}$ podemos encontrar fácilmente $\mathbb{P}$.
4) Encuentre el síndrome del vector recibido R = [1101011]. ¿Qué conclusiones puede extraer respecto de la secuencia enviada?
$*$ si hacemos $R\mathbb{H}^T$ obtenemos el síndrome $s = [1, 1, 1]$.
De [1] sabemos que el síndrome correspondiente a patrones de errores de un solo bit corresponde a las columnas de la matriz $\mathbb{H}$. Más específicamente, se puede probar que el síndrome de una palabra de código recibida es un vector que es igual a la suma de las columnas de H que corresponden a posiciones donde ocurrieron errores. Si desarrollamos un sistema de ecuaciones, llegamos a que hay y 2^k patrones de error que conducen al mismo síndrome. Sin embargo, se asume que el patrón más probable es aquel con error de menor peso. En este caso, el error más probable es haber fallado en el primer bit, por lo que la palabra que se intentó transmitir fue [0101011] que pertenece a C.
5) Bosqueje una implementación como un filtro de este código.
$*$ Veamos cuál es la salida de C para una secuencia arbitraria de 4 bits a la entrada.

$$\begin{pmatrix} x_1 & x_2 & x_3 & x_4 \end{pmatrix} \mathbb{G} =$$

$$\begin{pmatrix} x_2 + x_4 & x_1 + x_2 + x_4 & x_2 + x_3 & x_1 & x_1 + x_2 & x_3 & x_4 \end{pmatrix}$$

El diagrama de bloques correspondiente se observa en la figura 13.

B. Ejercicio 9.6

• Determine el código convolucional definido por la función de transferencia en transformada $D = [1, 1 + D + D^2, 1 + D^2]$

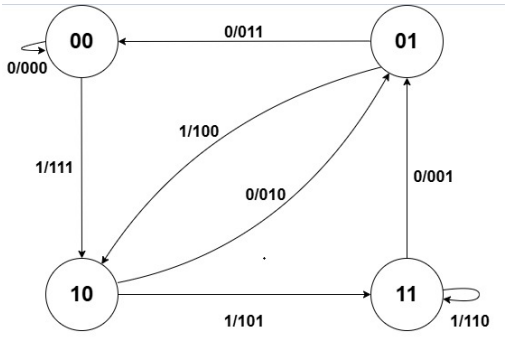

Fig. 14. Diagrama de estados

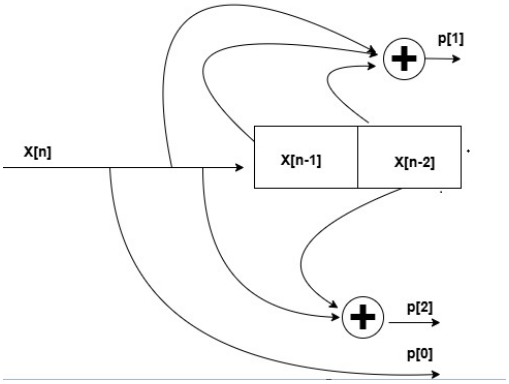

Fig. 15. Diagrama shift-register

1) Dibuje un diagrama tipo shift-register y un diagrama de estados.

∗ El diagrama de tipo shift-register se observa en la figura 15 y el diagrama de estados en la figura 14. Las sumas en el diagrama del shift-register son en módulo 2.

2) Dibuje un diagrama de Trellis de 4 transiciones de estados del código anterior.

∗ El diagrama de Trellis se observa en la figura 16.

3) Utilizando el algoritmo de Viterbi encuentre la secuencia de entrada más probable si la secuencia de salida recibida es 111,100,001,011.

∗ En la figura 17 se puede observar el algoritmo de Viterbi

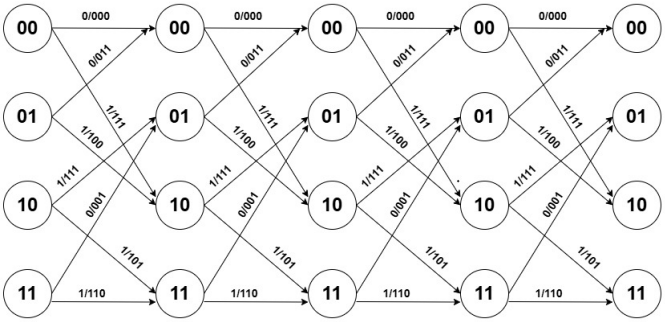

Fig. 16. Diagrama de Trellis

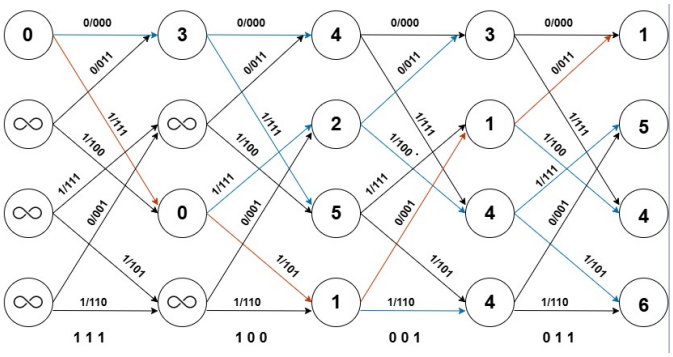

Fig. 17. Algoritmo de Viterbi

aplicado a la secuencia de entrada para este código en particular. La secuencia de entrada más probable se obtiene siguiendo el camino de las flechas anaranjadas con peso total de 1. El error se encuentra cuando se recibe 1 0 0 en vez de 1 0 1, y la secuencia que se envió es 1 1 0 0. Las flechas de color celeste indican el camino que se recorre para alcanzar el peso que figura dentro de cada nodo.

C. Ejercicio 9.7

1) Analizar el archivo ejercicio 7 a.grc y calcular el BER para los valores de SNR= 5,4,3,2,1,0,-1,-2,-3,-4,-5.

2) Analizar el archivo ejercicio 7 b.grc y calcular el BER para los valores de SNR= 5,4,3,2,1,0,-1,-2,-3,-4,-5. Graficar el BER de ambas partes y comparar.

∗ En las tablas III y IV se pueden apreciar los valores obtenidos para el BER en función del SNR de cada caso, respectivamente. En la figura 18 se puede ver una gráfica de la comparación entre ambas tablas. Se aclara que para el caso de la tabla IV en la celda en que hay un rango de -6 a -26 se tomó el punto medio entre estos dos valores (-16) para graficar.

Observando la figura 18 se aprecia como en ambos casos el BER disminuye a medida que se aumenta el SNR. Sin embargo, existe una notable diferencia de decrecimiento para el caso 2 a partir de un SNR de 3dB en adelante. Esto se debe a que en el caso 2, el sistema cuenta con bloques especiales que implementan corrección de error con diagramas de Trellis.

Para valores de SNR menores a 0dB ambos sistemas tienen comportamientos similares.

3) Analizar el archivo ejercicio 7 c.grc, y utilizar un archivo WAV que tenga en su equipo. Aumentar la potencia de ruido hasta el punto en que se comienza a sentir ruido y registrar estos dBs. Modificar el archivo anterior para utilizar un codificador de Trellis y decodificador de Viterbi Combo. Volver a calcular ahora el punto en el que comienza a sentirse ruido y registrar los dbs. ¿Cuál es la ganancia en dbs en la calidad percibida del audio por utilizar un código de corrección de errores?

SNR	BER
-5	-0.4
-4	-0.46
-3	-0.52
-2	-0.6
-1	-0.73
0	-0.965
1	-1.27
2	-1.71
3	-2.47
4	-3.43
5	-4.6

TABLE III
TABLA A: RELACIÓN SNR VS BER

SNR	BER
-5	-0.5
-4	-0.6
-3	-0.73
-2	-0.96
-1	-1.34
0	-1.93
1	-2.8
2	-4.2
3	-6.5
4	-26
5	-37.93

TABLE IV
TABLA B: RELACIÓN SNR VS BER

$*$ Al realizar el experimento para la canción ”Hung Up” de Madonna, para un volumen de 0dB se considera que existe ruido en el sonido a partir de una potencia de ruido de -3.8dB.

Lamentablemente, no fue posible reproducir el experimento utilizando el bloque de corrección de errores Trellis/Viterbi. Sin embargo, lo que se esperaría observar es que, al incorporar dicho bloque, el sistema adquiera una cierta inmunidad frente al ruido. En consecuencia, sería necesaria una potencia de error significativamente mayor para que comiencen a notarse distorsiones en el audio.

V. REFERENCIAS

[1] P. Belzarena & F. Larroca, ”Comunicaciones inalámbricas: Notas del curso”, Facultad de Ingeniería, Universidad de la República,

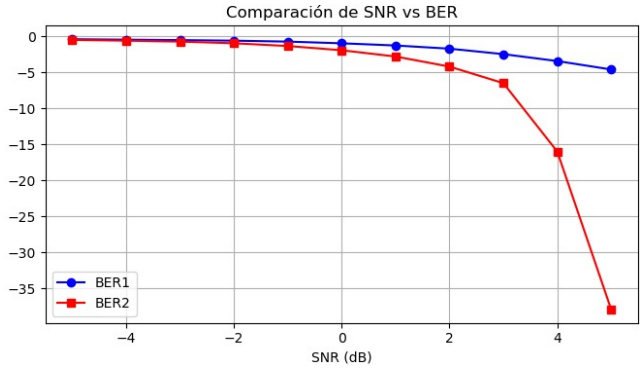

Fig. 18. Gráfica de SNR vs BER para ambos casos