Trabajo Práctico Número 2

Grupo número 8

UNMdP – Facultad de Ingeniería

Materia: Teoría de la Información

Profesores:

- Massa, Stella Maris
- Spinelli, Adolfo Tomás
- Lanzilotta, Franco

Autores:

- Battista, Franco *(franco.battista99@outlook.com)*
- Boolls, Nicolás *(nicolasboolls@gmail.com)*
- Mujica, Juan Manuel *(mujicajuanm@gmail.com)*

Fecha de entrega: 22 de noviembre de 2022.

Github: https://github.com/JuanMaMujica/TeoriaDeLaInformacion

Índice

Objetivos

- Obtener las codificaciones Huffman y Shannon-Fano para cada archivo.
- Comprimir cada archivo a partir de las mismas y además mediante el algoritmo RLC.
- Calcular y comparar las tasas de compresión entre los métodos para archivos de texto e imagen y extraer conclusiones.
- Analizar los distintos canales de información.
- Describir las características y propiedades de los mismos.

Resumen

El trabajo trata temas referentes a la codificación de fuentes de información, compresión de datos y canales de información. A lo largo del trabajo se cubren los temas enumerados a continuación:

- **Codificación de fuentes de información:** Primer Teorema de Shannon; Rendimiento y redundancia de un código; Códigos de Huffman y Shannon-Fano; Compresión de datos; Métodos de compresión (con pérdida o sin pérdida); Run Lenght Coding (RLC) .

- **Canales para la transmisión de información:** Medios de transmisión; Canales de información; Probabilidades asociadas a un canal; Entropías “a-priori” y “a-posteriori”; Equivocación de un canal; Información mutua; Propiedades de la Información mutua.

Introducción

Para el desarrollo del trabajo práctico los dos aspectos principales a considerar fueron: la resolución de los ejercicios, cada uno con su propia complejidad respecto de los temas tratados en las clases teóricas y prácticas, además del desarrollo de programas para cumplir los objetivos y obtención de resultados. Se utilizó el lenguaje Java para la codificación de los antes mencionados a pedido de la cátedra. Dicho lenguaje provee clases intrínsecas que facilitan el manejo de funciones con archivos (crear, abrir, modificar, leer, etc.), lo cual es útil, ya que a lo largo del trabajo se aprovecharon constantemente principalmente en la lectura y escritura de los mismos.

Parte A

Utilizando los distintos métodos mencionados en los objetivos se realizó la compresión de los archivos de texto e imagen brindados y se obtuvieron los siguientes resultados:

RENDIMIENTO	**Hr(S)/L**
REDUNDANCIA	**1 − (Hr(S)/L)**
TASA DE COMPRESIÓN N:1	**N = Tamaño original / Tamaño comprimido**

Propiedad	Huffman	Shannon-Fano
Rendimiento	**0.9899543475936531**	**0.9877849529886453**
Redundancia	**0.010045652406346917**	**0.012215047011354674**
Tasa de compresión	**1.9059881884356544:1**	**1.869092983456931:1**

Método Huffman

Los códigos de Huffman presentan un mayor rendimiento (por lo tanto menor redundancia) y mayor tasa de compresión (exceptuando un caso particular) que los valores obtenidos por los demás algoritmos. Un factor es que las longitudes medias de los códigos resultantes por Huffman son menores con respecto a las de los otros métodos y lo que resulta, en parte, que las tasas de compresión sean mayores para este, salvo por la imagen que se desarrollará en el análisis general.

Pseudocódigo:

```
procedimiento huffman(arbol a, char binario[], int longitud)
        char auxBinario[20]=""
        si (a) entonces
                si (a->padre!=null) entonces
                        strcpy(auxBinario,a->padre->codigoBinario)
                        strcat(auxBinario,binario)
                        strcpy(a->codigoBinario,auxBinario)
                fin si
                huffman(a->izq,"0",longitud)
```

```
                    si strlen(a->simbolo)==longitud entonces
                        escribir("Simbolo: %s / codigo Huffman: %s\n", a->simbolo,
a->codigoBinario)
                    fin si
                        huffman(a->der,"1",longitud)
```

Para armar el árbol de huffman hay un procedimiento previo en donde se van fusionando 2 nodos con la menor probabilidad y así sucesivamente hasta que quede un único nodo con probabilidad 1. Al ser extensa la implementación se decidió mostrar el procedimiento en donde se obtiene la codificación de Huffman para cada símbolo. Cada nodo hijo cuenta con un puntero al "padre" para así facilitar la codificación de huffman para cada símbolo, luego por recursividad se llega a cada uno de los nodos mediante un recorrido inorden.

COMPRESIÓN DE LA TABLA : CONVENCIONES ADOPTADAS

Para lograr la compresión por medio de huffman, es necesario tomar convenciones respecto a cómo guardar la tabla (generada por el árbol de huffman), junto con los códigos que luego ayudarán a lograr la descompresión si así se quiere hacer.

La estructura para comprimir la tabla que se usó fue la siguiente

- Primer byte del archivo, corresponde a la longitud de la tabla, es decir, la cantidad de símbolos distintos que había en el archivo original
- Luego, empezaría la tabla, recorriendose la cantidad leída del primer byte del archivo:
    - 1 byte correspondiente al símbolo, en valor ASCII
    - 1 byte correspondiente a la longitud del código huffman asociado a ese símbolo
    - n bytes correspondientes a en cuántos bytes puede almacenar efectivamente el código.

Quedando entonces, de la siguiente manera:

SÍMBOLO ASCII	LONGITUD CÓDIGO	CÓDIGO HUFFMAN
1 Byte	1 Byte	n bytes

Para la descompresión, no fue más desafío que leer el archivo con las convenciones tomadas, y luego rearmar la tabla para tener los códigos correspondientes. Luego, se recorrió el archivo bit a bit y se fue buscando los códigos que correspondieran al carácter leído para así imprimirlo.

Método Shannon-Fano

Para la implementación del algoritmo de Shannon-Fano utilizamos tres colecciones donde fuimos almacenando en cada una de ellas: todos los caracteres del archivo a comprimir, sus frecuencias y por último un objeto SimboloShannon que contiene un símbolo y su codificación correspondiente. Esta última colección es la que utilizaremos para realizar la compresión y la denominamos como “List simbolos”. Una vez armado nuestra colección de SimboloShannon y haciendo uso de un archivo de texto auxiliar pudimos comprimir correctamente nuestro archivo y nuestra tabla utilizando la misma convención mencionada anteriormente para el algoritmo de Huffman.

Pseudocódigo para el método Shannon-Fano:

```
procedimiento iniciaShannon(int limiteInf, int limiteSup, List simbolos)
        si limiteInf != limiteSup entonces
                int i, j, corte <- 0 float inf <- 0, sup <- 0, dif <- 0, probDif <- 0
                para i = limiteInf hasta i <= limiteSup con pasos de 1 hacer
                        sup <- sup + simbolos.get(i).getProbabilidad()
                        para j <- i + 1 hasta j <= limiteSup con pasos de 1 hacer
                                inf <- inf+ simbolos.get(j).getProbabilidad()
                        fin para
                        dif <- (sup - inf) * (sup - inf)
                        si i == limiteInf entonces
                                probDif <- dif
                        sino si dif >= probDif entonces
                                i <- i - 1
                        fin si
                        probDif <- dif
                        fin si
                        inf <- 0
                fin para
                corte <- i
                shannonFano(limiteInf, limiteSup, corte, simbolos)
                iniciaShannon(limiteInf, i, simbolos)
                iniciaShannon(i + 1, limiteSup, simbolos)
        fin si
fin procedimiento
```

El procedimiento de Shannon Fano busca formar subconjuntos cuya diferencia entre la sumatoria de las probabilidades de los símbolos de dichos subconjuntos en valor absoluto sea mínima. El primer procedimiento arma los subconjuntos, y el otro procedimiento que se encuentra a continuación va designando el bit correspondiente (0 o 1) para cada símbolo según en cual subconjunto se encuentre.

```
procedimiento shannonFano(int limiteInf, int limiteSup, int corte, List simbolos)
int i
 para i = limiteInf hasta i <= limiteSup con pasos de 1 hacer
        si i <= corte entonces
                simbolos.get(i).setCodigoShannon(simbolos.get(i).getCodigoShannon() + "1");
```

```
        sino
                simbolos.get(i).setCodigoShannon(simbolos.get(i).getCodigoShannon() + "0");
        fin si
fin para
fin procedimiento
```

Como es de esperarse, el rendimiento al utilizar el algoritmo de Huffman fue ligeramente menor que utilizando el de Shannon Fano. Esto corresponde al temario visto en la teoría, donde se aclara que los resultados de utilizar Huffman serán óptimos a diferencia de su contraparte que a pesar de tener valores de rendimiento muy similares, no alcanzan dicha optimidad. Además de observar estos resultados en los rendimientos y redundancias, terminamos de comprender la comprensión analizando el tamaño de los archivos comprimidos, quedándonos con huffman una tasa de compresión de 1.9059881884356544:1 y con Shannon Fano una tasa de compresión de 1.869092983456931:1
Notamos también que hay una ligera diferencia en las longitudes medias (diferencia que lleva a Huffman a ser más óptimo a fin de cuentas), pero es interesante analizar que mientras que para Shannon Fano la longitud media de las palabras código es de aproximadamente 4.262 bits/símbolo, para Huffman es de aproximadamente 4.255 bits/símbolo. Este es el ejemplo más crudo para comprender por qué un método es ligeramente más eficiente que otro. Mientras que el algoritmo de Huffman utiliza menor cantidad de bits para definir todas sus palabras código y se implementa con cierta dificultad, el algoritmo de Shannon Fano utiliza una cantidad ligeramente mayor, aunque se implementa con mayor facilidad.

Parte B

La segunda parte del trabajo práctico consistió en el análisis de 3 canales de comunicación diferentes, de los cuáles tuvimos que calcular su equivocación, la información mutua y las propiedades de cada canal.
Comenzamos calculando los valores faltantes de la matriz de canal (véase las matrices completas en el anexo). Las incógnitas nos quedan de la siguiente manera:

	Canal 1	Canal 2	Canal 3
a	0.24	0.24	0.24
b	0.32	0.16	0.16

c	**-**	**0.24**	**0.24**

Una vez tenemos la matriz de canal completa podemos obtener las probabilidades del alfabeto de salida a través de la siguiente ecuación:

$$P(bj)=\sum_{i=1}^{r} P(ai)\,Pij \qquad \forall\, j=1..s$$

Tomaremos el Canal 1 como ejemplo, se pueden observar las probabilidades de los alfabetos de salida del canal 2 y 3 en la hoja de cálculos:

Símbolo	**P(bi)**
B1	**0.284**
B2	**0.312**
B3	**0.404**

Luego se calculan las entropías a posteriori para el cálculo de la equivocación:

$$H(A/bj)=\sum_{A} P(a/bj)\log\frac{1}{P(a/bj)}$$

Estas no quedan de la siguiente manera para el canal 1 (tanto para el canal 2 y 3 se verán dichas tablas en el excel):

H(A/B1)	**1,941925754 bits**
H(A/B2)	**1,911683085 bits**
H(A/B3)	**1,978539355 bits**
H(A/B4)	**1,889729817 bits**

Calculamos la entropía a priori para saber la información mutua con la siguiente ecuación:

$$H(A)=\sum_{A} P(a)\log\frac{1}{P(a)}$$

Dándonos como resultado:

H(A)	**1,948 bits**

Finalmente calculamos la equivocación y la información mutua:

Equivocación:

$$H(A/B)=\sum_{B} P(b)\,H(A/b)=\sum_{A,B} P(a,b)\log\left(\frac{1}{P(a/b)}\right)$$

Información Mutua:

$$I(A,B)= H(A)-H(A/B)$$

Se requería también verificar las propiedades de la información mutua (para cada canal), las cuales son:

- I (A,B) ≥ 0 : No se pierde en absoluto información por el hecho de observar la salida del canal. Además, la condición para que la información mutua sea nula es que los símbolos de entrada y salida sean estadísticamente independientes, lo cual de antemano se sabe que no lo son (ver apéndice con las probabilidades)
- I (A,B) es simétrica respecto a las variables ai y bj. Por lo tanto, podrá escribirse:
    - I (A,B) = I (B,A) (reciprocidad de la información mutua).
    - Luego:
        - I (A,B) = H (B) - H (B/A) ○ H (A,B) = H (A) + H (B) - I (A,B)

Los resultados pertinentes a los tres canales son los siguientes:

	H(A/B)	H(B)	H(B/A)	I(B/A)	I(A,B)	H(A)
Canal 1	2,149 bits	1,568 bits	1,546 bits	0,022 bits	0,022 bits	2,171 bits
Canal 2	1,923 bits	1,963 bits	1,938 bits	0,025 bits	0,025 bits	1,948 bits
Canal 3	2,502 bits	1,973 bits	1,948 bits	0,025 bits	0,025 bits	2,527 bits

Algunas conclusiones que podemos sacar son las siguientes.
Un canal perfecto es aquel que dado el envío de un símbolo ai del alfabeto de entrada A, siempre se recibe el mismo símbolo del alfabeto de salida B. La matriz de un canal de ese tipo tiene todos sus elementos=0, salvo los elementos de la diagonal principal que deben ser unos. Por construcción, para que un canal sea perfecto la matriz del canal debe ser de NxN, es decir, los alfabetos A y B deben contener la misma cantidad de símbolos. Al no ser matrices cuadradas (ver apéndice) se descartan los canales 1 y 3 como canales perfectos. Asimismo, todas las matrices son distintas a una matriz identidad, es por ello que el canal 2 no es perfecto (ninguno de ellos lo es), sino que son canales con distorsiones.

La equivocación del canal o ruido mide la información que queda en A o la entrada después de observar B o la salida. La información mutua de un canal se define como la cantidad de información que se obtiene de A gracias al conocimiento de B o también como la cantidad de información sobre A que atraviesa el canal, esta se obtiene realizando la diferencia H(A) - H(A/B). Un canal es perfecto si el valor de su información mutua es equivalente a la entropía a priori y el valor de su equivocación es nula. Como se anticipó, ninguno de los canales es perfecto, por lo contrario, son bastante pobres porque hay ruido, como también se lo conoce a la equivocación de un canal, considerable en cada uno: El canal 1 es considerado el peor, ya que es el que presenta la menor información mutua, es decir, el canal con mayor distorsión en comparación a los demás y que por ello es el canal en el que menos información se obtiene de la entrada conociendo la salida del mismo, seguido del canal 2 y 3, que aún siendo canales pobres, son levemente mejores que el anterior. Sin embargo, ningún canal está cerca de cumplir la igualdad I (A,B) = H (A), ratificando la imperfección de ellos. La información mutua depende del canal,es decir, depende de la fuente que genera los símbolos de entrada del canal (sus probabilidades). Y depende de cómo se utilice: cada símbolo de 7 entradas tiene cierta probabilidad de entrar al canal, por lo que si se quisieran modificar las probabilidades entre los símbolos, todos los resultados serían, en principio, diferentes. Por otra parte, en todos los casos se da que H(A) > H (A/B), es decir, que en promedio nunca se pierde información al conocer la salida. Como se vio, hay canales que pierden más información que otros a causa de la equivocación pero nunca sucederá que la equivocación de un canal sea mayor que la entropía a priori.

Conclusión

Con respecto a los resultados de la primera parte podemos afirmar que efectivamente el método de compresión de Huffman es de mayor rendimiento y menor redundancia que el Shannon Fano. Esto se debe a que las longitudes medias de Huffman son menores que las de Shannon fano por lo que la tasa de compresión es mayor.
Nuestro mayor desafío al implementar los métodos de Huffman y Shannon Fano fue el determinar la forma en la que se comprimiría la tabla, y las convenciones que tomamos. Una vez determinada estas, el proceso se tornó mucho más sencillo y se logró la comprensión y la descompresión de forma correcta.

En la segunda parte del trabajo práctico pudimos obtener como conclusión que en todos los canales se presenta ruido, esto se debe a que las matrices son diferentes a la matriz identidad, la cual sería un caso ideal donde no se pierde información. Podemos concluir entonces que el canal 1 es el más pobre de los 3 ya que presenta mayor distorsión seguidos del canal 2 y 3 que a pesar de ser canales que también poseen ruido tienen un poco menos que el primero.
Otra de las conclusiones que sacamos es que en promedio en ninguno de los 3 canales se pierde información si se sabe la salida.

¡Gracias por leer!

Anexo

1. Canal 1

Canal de Entrada	
Simbolo	P(i)
S1	0,2
S2	0,1
S3	0,3
S4	0,3
S5	0,1

Matriz del canal			
	B1	B2	B3
S1	0,3	0,24	0,46
S2	0,32	0,4	0,28
S3	0,3	0,24	0,46
S4	0,24	0,4	0,36
S5	0,3	0,32	0,38

2. Canal 2

Canal de Entrada	
Simbolo	P(i)
S1	0,25
S2	0,33
S3	0,27
S4	0,15

Matriz del canal				
	B1	B2	B3	B4
S1	0,2	0,24	0,16	0,4
S2	0,24	0,3	0,16	0,3
S3	0,24	0,16	0,2	0,4
S4	0,24	0,3	0,24	0,22

3. Canal 3

Canal de Entrada	
Simbolo	P(i)
S1	0,15
S2	0,1
S3	0,2
S4	0,25
S5	0,14
S6	0,16

Matriz del canal				
	B1	B2	B3	B4
S1	0,2	0,24	0,16	0,4
S2	0,24	0,24	0,3	0,22
S3	0,16	0,2	0,24	0,4
S4	0,24	0,3	0,16	0,3
S5	0,2	0,24	0,24	0,32
S6	0,16	0,24	0,3	0,3