Trabajo Integrador Nº2

Integrantes: Joaquin Fioriti

Mariano Garcia

Ramiro Pruis

Repositorio GitHub

Fecha de entrega: 21/11/21

Teoría de la Información

Email de contacto: joaquinfioriti9@gmail.com

Resumen **3**

Introducción **3**

Desarrollo **3**

Primer Parte 3

Segunda Parte 8

Conclusión **10**

Anexos **10**

Anexo A 10

Anexo B 11

Resumen

El trabajo consiste en aplicar distintos métodos de compresión (RLC, Shanon-Fano y Huffman) sobre varios archivos dados. Con los archivos codificados resultantes, se realizan diferentes cálculos y se comparan y se interpretan los resultados.

Luego a partir de los canales de comunicación dados, se calcula Equivocación, información Mutua y propiedades de cada canal. A partir de estos resultados se realizan comparaciones y conclusiones.

Introducción

La compresión de datos es sumamente necesaria en nuestros días. Grandes cantidades de información son transmitidas y guardadas en cada momento y, de hacer esto utilizando el tamaño normal de esos archivos, podría saturarse el sistema. Por eso, a lo largo del tiempo, y utilizando distintas técnicas basadas en modelos estadísticos, algebráicos, algorítmicos, se han inventado diversas formas de compresión para acelerar los procesos de transmisión de información y disminuir también el tamaño de los archivos contenedores de esa información. Bajo el caso de estudio actual, se informará sobre los algoritmos de compresión de Huffman, Shannon-Fano, RLC. Utilizando los archivos obtenidos luego de utilizar los distintos métodos se documentará la tasa de compresión, el rendimiento y la redundancia.

En la segunda parte del trabajo se propone trabajar con los canales para la transmisión de la información. Se utilizarán las matrices de los respectivos canales para así obtener las entropías que servirán para calcular la equivocación de un canal, la información mutua del mismo. También, de esta última se verificarán las propiedades que tiene.

Desarrollo

Primer Parte

En esta primera parte, se dispone de tres archivos diferentes: : *Argentina.txt, Hungaro.txt* e *Imagen.raw.* El primero contiene el Himno Nacional de Argentina, el segundo tiene el mismo himno, pero traducido mediante Google Translator. Por último, imagen.raw contiene una imagen del personaje clásico “Mafalda”.

Como cualquier archivo, estos tienen su tamaño. Se solicita, mediante diferentes algoritmos vistos a lo largo de la materia, comprimirlos a cada uno y, a partir de los resultados obtenidos, plantear conclusiones y comparaciones desde el punto de vista de la compresión, el rendimiento y la redundancia.

Los algoritmos de compresión desarrollados fueron los siguientes:

- Huffman
- Shannon-Fano
- RLC

Algoritmo de Huffman

_____El algoritmo de Huffman propone desarrollar un código instantáneo y óptimo a partir de una fuente con memoria nula. Este consiste en la creación de un árbol binario donde se marcan los nodos hojas con los caracteres en conjunto a sus frecuencias, de este modo se va uniendo cada pareja de nodos de menor frecuencia, creando un nuevo nodo etiquetado con dicha suma. Estos pasos se repiten hasta que no queden nodos hojas por unir a ningún nodo superior.

Luego se etiquetan las aristas que unen estos nodos con ceros y unos. Por lo que la creación de los nuevos símbolos va a ser dada en el recorrido de este árbol hasta encontrar la posición del símbolo original.

La eficiencia de este algoritmo se puede analizar claramente en las aplicaciones propuestas en el trabajo. Una vez realizada la compresión de los tres archivos y analizando en consecuencia al teorema de codificación de fuente de Shannon, se obtuvieron los siguientes resultados.

Archivo	H(A)	L(A)
Argentina.huf	7.57	7.61
Hungaro.huf	7.63	7.71
Imagen.huf	1.07	1.37

El primer teorema de Shannon nos asegura que es imposible comprimir la información de forma que la cantidad de bits por símbolo sea menor que la entropía de Shannon, asegurando que no haya pérdida de información. Por lo que comparando los resultados dados, no solo vemos que este teorema se cumple, sino que los resultados experimentales son muy cercanos al valor teórico dado por Shannon, por lo que concluimos que el algoritmo de Huffman es óptimo.

Algoritmo de Shannon-Fano

_____El algoritmo de Shannon Fano es un procedimiento subóptimo para construir un código que alcanza una cota de $L \leq H(S) + 2$. En este algoritmo, se parte de una lista que vincula el valor del símbolo con su probabilidad. Está ordenada descendentemente a partir de este último criterio. De esta lista se generan dos sublistas de símbolos cuyas frecuencias sumadas sean similares, es decir, la suma de frecuencias de la primera sublista debe ser similar a la suma de frecuencias de la segunda. Los símbolos de la primera lista recibirán el “0” y los de la segunda el “1”. Este proceso se sigue haciendo para cada sublista mientras haya más de un término en la lista actual. De esta manera, cada símbolo va generando un número binario dependiendo en que sublista le toque en cada iteración. Por último cuando todas las sublistas sean de un elemento, cada símbolo tendrá asociado un número binario que lo represente. Como aquellos símbolos que más frecuencia tienen siempre están en la

primera sublista, contienen menos “1”, y si los contienen son al final, es decir, los símbolos con más frecuencias son representados con números binarios más pequeños y, por lo tanto, de menos tamaño. Esta estrategia plantea que si hay símbolos que tienen alta frecuencia, que ocupen lo menos posible.

Los pasos entonces son:

- Para una lista de símbolos dada, crear su correspondiente lista de probabilidades o de frecuencias de aparición de manera que se conozca la frecuencia relativa de ocurrencia de cada símbolo.
- Ordenar las listas de símbolos de acuerdo a la frecuencia, con los símbolos de ocurrencia más frecuente a la izquierda y los menos comunes a la derecha.
- Dividir la lista en dos partes, haciendo la frecuencia total de la mitad izquierda lo más próxima posible a la de la mitad derecha.
- Asignar a la mitad izquierda el dígito binario “0”, y a la mitad derecha el dígito “1”. Esto significa que los códigos para los símbolos en la primera mitad empezarán con “0”, y que los códigos de la segunda mitad empezarán por “1”.
- Aplicar recursivamente los pasos 3 y 4 a cada una de las dos mitades, sub dividiéndolas en grupos y añadiendo bits a los códigos hasta que cada símbolo se corresponde con una hoja del árbol.

```
proceso shannonFano(resultado[caracter, cadenaBits], listaCaracteres, boolean izq){
    si es parte de la lista de la izquierda:
        bit = “0”
    si no
        bit = “1”

    por cada caracter en la lista de caracteres:
        string = resultado.getCadenaFormadaHastaAhora
        resultado(caracter, string + bit)


    si el tamaño de la lista actual es mayor a 1:
        separador = encuentraSeparadorIgualProbabilidad(listaCaracteres)
        listaIzquierda = listaCaracteres.crearSubLista(0, separador)
        shannonFano(resultado, listaIzquierda, true)
        listaDerecha = listaCaracteres.crearSubLista(separador, listaCaracteres.tamaño())
        shannonFano(resultado,listaDerecha,false)
}
```

Algoritmo RLC

_____El algoritmo de RLC, equivalente a Run Length Coding por sus siglas es una forma simple de compresión de datos. Consiste en tomar secuencias consecutivas de datos del mismo valor y almacenar la cantidad de veces que se repitió la misma variable hasta que haya un cambio. El mismo sistema se aplica para todo el documento analizado. Es un algoritmo altamente eficiente para archivos con muchas secuencias de caracteres repetidas. Su implementación se realizó de la siguiente manera:

```
proceso escribeCodificadoRLC (palabra[], archivo archivo_a_grabar,
booleano imagen)
  char strout
  i = 1
  n = largo(palabra)
  contador = 0

  hacer
    si (palabra[i] = palabra [i-1])
       contador = contador + 1
    sino
       strout = palabra[i-1]
       imprimir(archivo_a_grabar,contador)
       imprimir(archivo_a_grabar,strout)
       contador = 1
    fin si
  i=i+1
  mientras (i < n)
fin proceso
```

De esta forma, se toma un caracter, y se itera en el texto hasta que el nuevo valor sea distinto del anterior. Cuando ocurre esto, se imprime en el archivo correspondiente el número de repeticiones del caracter y el caracter en sí.

_____Utilizando los tres algoritmos, se procedió a tomar cada uno de los ficheros otorgados, y generar los archivos comprimidos correspondientes a los mismos luego del uso del mismo por parte de cada método de compresión.

Antes de interpretar, comparar y concluir acerca de los resultados, vamos a definir los conceptos. Para ello suponemos algunos significados:

L = longitud media de un código

H = entropía de una fuente

S = una fuente

La **tasa de compresión** de un archivo es la división entre el tamaño original del archivo y el tamaño comprimido, y se expresa de la siguiente manera N:1, siendo N un natural.

$$N = tamOriginal / tamComprimido$$

El **rendimiento** o eficiencia de un código r-ario se define como:

$$\eta = H_r(S) / L$$

La **redundancia** se define como:

$$1 - \eta = (L - H_r(S)) / L$$

Una mayor redundancia significa menor información, es decir, innecesaria.

A continuación se muestra la tabla de resultados:

	Huffman		**Shannon-Fano**		**RLC**	
	Variable	**Valor**	**Variable**	**Valor**	**Variable**	**Valor**
	Compresión	6:1	Compresión	1:1	Compresión	0:1
Argentina.txt	Rendimiento	0.995	Rendimiento	0.983	Rendimiento	0.563
	Redundancia	0.004	Redundancia	0.016	Redundancia	0.436
	Compresión	7:1	Compresión	1:1	Compresión	0:1
Hungaro.txt	Rendimiento	0.990	Rendimiento	0.981	Rendimiento	0.307
	Redundancia	0.009	Redundancia	0.0186	Redundancia	0.693
	Compresión	17:1	Compresión	3:1	Compresión	17:1
Imagen.raw	Rendimiento	0.777	Rendimiento	0.913	Rendimiento	0.044
	Redundancia	0.222	Redundancia	0.086	Redundancia	0.956

Veamos qué quieren decir estos resultados. En un primer vistazo, se concluye que el algoritmo que más tasa de compresión logró, fue el de Huffman tanto para los archivos de texto como para la imagen. Esto es debido al análisis previamente realizado sobre la optimización que asegura este algoritmo.

Sobre el algoritmo de Shannon-Fano, vemos que las tasas de compresión no son elevadas, casi el mismo tamaño luego de aplicar el algoritmo. Sin embargo hay una notable

mejora en la comprensión de la imagen.raw, en la que se disminuye tres veces el tamaño del archivo original. Este algoritmo se destaca por su alto rendimiento en los tres casos

Respecto al rendimiento, la relación entre la longitud media y la entropía tiende a 1 en la mayoría de los casos, salvo para el caso de RLC, en donde vemos valores de rendimiento muy inferiores a los demás. Esto es así porque la longitud media que se obtiene para el RLC se encuentra fuera del intervalo establecido por el primer Teorema de Shannon. Como vamos a necesitar una cantidad mínima de bits para representar tanto el símbolo como la cantidad de apariciones, la longitud media va a ser la suma de estas dos. Por lo tanto, no tendremos un código compacto, ya que no es el de menor longitud media posible. En consecuencia, los resultados de rendimiento arrojados por el método de compresión RLC son muy bajos ya que la longitud media es mucho mayor a la entropía.

Por último, vemos las tasas de compresión del algoritmo RLC. En estos casos, vemos valores de "compresión" inusuales para el caso de los archivos de texto. Sin embargo en la imagen se obtuvo una tasa de compresión elevada en la que se redujo 17 veces el tamaño. Esto es así porque en los ficheros de texto no suele haber secuencias seguidas del mismo simbolo, por lo tanto, en promedio por cada letra tendremos un número asociado. En cambio, en la imagen se repiten grandes cantidades del mismo valor para poder representar al color, por lo que se toman todas estas, y el tamaño luego de la compresión disminuye notablemente.

Segunda Parte

En esta instancia de trabajo tendremos que analizar 3 canales de comunicación. Un canal de comunicación cuenta con un alfabeto de entrada A = {a1,a2,...,an}; un alfabeto de salida B = {b1,b2,...,bn}; y un conjunto de probabilidades condicionales P (bj/ai)

En cuanto los tres canales, a primera vista vemos que no son canales perfectos ya que en la matriz del canal vemos que para la salida *Si*, las probabilidades de obtener *Bi* están distribuidas. Por ende, no vamos a tener certeza sobre la entrada partiendo de la salida. Aunque, conociendo la distribución de probabilidades de la entrada P(ai), las probabilidades de la salida sabiendo la entrada P(bj,ai) y las probabilidades de salida P(bj), podemos conocer la probabilidad condicional de la entrada *ai* sabiendo la salida *bj* P(ai/bj)

Definiendo un canal de r símbolos de entrada y s de salida:

$$\sum_{j=1}^{s} Pij = 1 \, para \, i = 1 \ldots r \quad (1)$$

$$P(bj) = \sum_{i=1}^{r} P(ai) * Pij \; \forall \, j = 1 .. s \quad (2)$$

A partir de lo planteado y las ecuaciones (1) y (2) llegamos a la siguiente ecuación:

$$P(ai/bj) = \frac{P(bj/ai)*P(ai)}{P(bj)}$$ (ver Anexo A)

A esta probabilidad la llamamos con el nombre de probabilidad “a-posteriori”. Luego a partir de los postulados de Shannon podemos calcular la entropía a posteriori:

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

Siendo la cantidad media de binits para representar un símbolo de una fuente con probabilidad P(ai/bj).

A partir de esta entropía calculamos la Equivocación de A con respecto a B:

$$H(A/B) = \sum_{B} P(b)\, H(A/b)$$

Esto se interpreta como la pérdida de información sobre A provocada por el canal. También puede verse como el número mínimo de preguntas binarias en promedio para determinar la entrada, conociendo la salida. Por otro lado podemos definir H(B/A) como el mínimo de preguntas binarias en promedio para determinar la salida conocida la entrada, también conocido como pérdida.

Finalmente llegamos a la información mutua, siendo la cantidad de información de un símbolo de salida.

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

Definiendo estos conceptos, tomando los símbolos de S como entrada y B como salida, y aplicando los cálculos a los tres canales dados se obtuvieron los siguientes resultados:

	Equivocación H(S/B)	H(S)	Información Mutua I(S,B)
Canal 1	2.12	2.16	0.04
Canal 2	1,93	1.97	0.04
Canal 3	2.23	2.27	0.04

En estos resultados se puede ver que la entropía de la entrada H(S) es mayor a la equivocación, por lo que podríamos afirmar que en promedio no se pierde información al conocer la salida. Vemos que en todos los canales el valor de la información mutua es igual, esto quiere decir que la cantidad de información cuando *Si* es trasmitido y se recibe *Bi* es igual en todos los canales.

En cuanto a las propiedades de la información mutua vemos que para todos los canales $I(S,B) \geq 0$ por lo que no se pierde información al observar la salida de los canales. Además, realizando los cálculos necesarios (entropía de B, pérdida, e información mutua I(B,S)) *(ver Anexo B)*, se llegó a que los tres canales cumplen con la propiedad simétrica siendo $I(S,B) = I(B,S)$.

Conclusión

Durante el desarrollo del proyecto vimos la importancia de la compresión de la información y el hecho de cómo se realiza la misma. Entendiendo que a través de propiedades y postulados se desarrollan algoritmos de compresión utilizados hoy en día que permiten el flujo de información de manera compacta y correcta. Se utilizaron tres técnicas de compresión distintas para evaluar su comportamiento frente a distintos tipos de archivos y determinar así las ventajas y desventajas de los mismos en diferentes escenarios. Se cumplieron los objetivos dados y tanto la información brindada como los cálculos realizados son coherentes entre sí.

Mediante la utilización de un programa java, fue posible la resolución de los enunciados relacionados a la primera parte, cuyas respuestas fueron escritas en distintos archivos dentro de la carpeta “Resultados”. Luego, en relación a la segunda parte fue necesario la utilización de una hoja de cálculos.

En la segunda instancia de este trabajo, se analizaron todos los aspectos vistos sobre los canales de comunicación y en base a sus probabilidades y los postulados propuestos por Shannon, vimos y analizamos su comportamiento en la transmisión de información.

Anexos

Anexo A

Canal 1 - Matriz P(si/bj)

	S1	**S2**	**S3**	**S4**	**S5**
B1	0,340909	0,090909	0,227273	0,284091	0,056818182
B2	0,267857	0,238095	0,178571	0,297619	0,017857143
B3	0,3	0,24	0,2	0,1875	0,0725

Canal 2 - Matriz P(si/bj)

	S1	**S2**	**S3**	**S4**
B1	0,222222222	0,333333	0,222222	0,222222
B2	0,321428571	0,321429	0,142857	0,214286
B3	0,272727273	0,272727	0,181818	0,272727
B4	0,391304348	0,26087	0,26087	0,086957

Canal 3 - Matriz P(si/bj)

	S1	**S2**	**S3**	**S4**	**S5**	**S6**
B1	0,24	0,24	0,08	0,36	0,04	0,04
B2	0,310345	0,206897	0,068966	0,310345	0,051724138	0,051724
B3	0,25	0,25	0,125	0,25	0,0625	0,0625
B4	0,409091	0,090909	0,136364	0,272727	0,045454545	0,045455

Anexo B

Canales	H(B/S)	H(B)	I(B,S)
Canal 1	1,52	1,56	0,04
Canal 2	1,95	1,99	0,04
Canal 3	2,23	2,27	0,04

**Valores redondeados a 2 decimales*