TEORÍA DE LA INFORMACIÓN

Trabajo Práctico Integrador N°2

Grupo 9

Integrantes:

Ignacio Suárez Quilis

Lautaro Nahuel Bruses

Matías Angélico

E-mail de contacto:

ignaciogustavosuarez1234@gmail.com

lautarobruses@gmail.com

mnangelico@gmail.com

GitHub:
https://github.com/MouGliXx/teoria-de-la-informacion

1. Índice

2. Resumen

El trabajo se enfocó en aplicar los métodos de compresión de Huffman y Shannon-Fano sobre un archivo cedido por la cátedra, utilizando un diccionario de palabras como las palabras origen a comprimir. Se generó un archivo de salida por cada compresión resultante de cada algoritmo y se realizaron los cálculos pertinentes para su posterior comparación.

Luego el trabajo se centró en el estudio de diferentes canales de transmisión de información. A partir del cálculo de un conjunto de probabilidades condicionales que determinan una matriz, un canal queda completamente definido. En base a ésta, se calcularon las relaciones entre las probabilidades de cada canal, para de esta manera, poder obtener las entropías asociadas antes y después de entrar en dicho canal.

Por último, a partir de los datos obtenidos anteriormente, se calculó la incidencia del canal en los alfabetos de entrada y salida, calculando la equivocación, la información mutua, la entropía afín y la pérdida de cada uno de los respectivos canales.

3. Introducción

La compresión de datos es la reducción del volumen de datos tratables para representar una determinada información empleando una menor cantidad de espacio. Esta técnica se suele emplear para ahorrar espacio al guardar la información o para ahorrar tiempo al transmitirla. Al encargado de codificar la información se le llama “Compresor”, en cambio al encargado de decodificar el código para así obtener el mensaje original (o al menos una aproximación de este) se le llama “Descompresor”.

Existen varios métodos de compresión, los cuales se clasifican en “Sin pérdida o reversibles”, los cuales mantienen la integridad de la información, o “Con pérdida o irreversibles”, los cuales no mantienen esta integridad. En este trabajo se evaluarán con detalle dos métodos sin pérdida como lo son Huffman y Shannon-Fano.

Como la información se codifica antes del ingreso al medio (canal) y se decodifica a la salida, teniendo presente la posible existencia de ruido que perturbe esta transmisión, en la segunda parte del trabajo se estudió la transmisión/procesamiento de dicha información en los canales presentados por la cátedra, teniendo en cuenta la posibilidad de encontrar símbolos de distinta naturaleza en el centro emisor y en el centro receptor, así como también la posibilidad de aparición de señales incontrolables y aleatorias que modifiquen el valor del símbolo transmitido produciéndose errores.

4. Desarrollo

4.1 Primera parte

Inicialmente, en base al archivo de texto facilitado por la cátedra, se recolectaron todos los conjuntos de caracteres separados por espacios (no discriminando entre caracteres y caracteres especiales) y se generó un diccionario de palabras. A partir de este diccionario, se calculó la frecuencia de aparición asociada a cada palabra dentro del texto. Se calculó la respectiva información de cada palabra para, junto con las frecuencias obtenidas anteriormente, poder calcular la entropía de la fuente de información.

4.1.1 Algoritmo de Huffman

El algoritmo de Huffman es una de las técnicas de compresión vistas en el curso, el mismo busca reducir la cantidad de espacio necesaria para almacenar un mensaje. Utilizaremos una tabla que contendrá el diccionario, este nos permitirá saber las palabras que estarán presentes junto a su frecuencia.

Para la implementación del algoritmo, trabajaremos junto a la siguiente estructura:

```
class Nodo {
  String cadena;
  Integer frecuencia;
  Nodo izq = null, der = null;
}
```

El algoritmo se basa en el uso de una cola de prioridad con un comparador que permite saber qué nodo tiene menor frecuencia. Se pasaron los datos por parámetro, que contendrán las palabras con sus respectivas frecuencias y se conformó una cola con las cadenas ordenadas por frecuencia. Se iteró mientras la longitud de la cola sea mayor a 1, en cada ciclo se sacara los dos últimos nodos que serán los de menor frecuencia y se insertará un único nodo con la frecuencia resultado de la suma, teniendo como hijos a izquierda y derecha los nodos extraídos previamente de la cola. Luego, con la función encode se recorre el árbol concatenando dentro de un parámetro llamado “**str**”: 0 o 1, al

llegar a una hoja la recursión se detiene. Es así como se obtuvo una codificación que relaciona el árbol de Huffman con la original. Una vez realizado esto, se almacenaron los códigos obtenidos en el archivo comprimido "*Huffman.Huf*". Y se almacenó el diccionario de codificación de dicho algoritmo, junto con los resultados de los cálculos en el archivo de texto "*Huffman.txt*".

4.1.2 Algoritmo de Shannon-Fano

Es una técnica utilizada en la compresión de datos para construir un código prefijo basado en un conjunto de símbolos y sus probabilidades que alcanza una cota de $L \leq H(S) + 2$.

Este algoritmo parte de una secuencia de símbolos, en base a la cual se genera una lista con las frecuencias de aparición correspondientes a cada símbolo y se la ordena de forma decreciente. Esta lista se divide en 2 sublistas (superior e inferior), de forma que la diferencia entre las sumas de las frecuencias entre ambas sublistas sea la mínima posible. A la sublista superior se le asigna el dígito binario **'0'**, y la mitad inferior, el dígito binario **'1'**. Esto provocará que todas las codificaciones de los símbolos pertenecientes a la mitad superior comiencen con **'0'**, y las codificaciones de los símbolos de la mitad inferior comiencen con **'1'**. Este proceso se repetirá recursivamente, agregando dígitos binarios a las codificaciones, hasta que cada sublista conste de un único símbolo.

El algoritmo funciona, y produce codificaciones de longitud variable bastante eficientes cuando las dos sublistas producidos por una división tienen la misma probabilidad, ya que el bit de información usado para distinguirlos se usa más eficientemente.

Se utilizó este algoritmo para comprimir el archivo provisto por la cátedra. A partir de las frecuencias calculadas anteriormente, se las ordenó de forma decreciente para poder ingresarlas en el método recursivo principal encargado de generar las codificaciones de Shannon-Fano de cada código.

```
frecuencias -> Mapa<codigo, frecuencia>
codificacionActual-> String
shannonFano -> Mapa<codigo, codificacion>

generaShannonFano(frecuencias, codificacionActual) {
   k = 0
   if (tamano(frecuencias) == 1) {
      if (longitud(codificacionActual) > 0) {
         put (codigo, codificacionActual) in shannonFano
      } else {
         put (codigo, '1') in shannonFano
      }
   } else {
      k = calculaK(frecuencias)

      mitadSuperior = cortaMitadSuperior(frecuencias, k)
      mitadInferior = cortaMitadInferior(frecuencias, k)

      generaShannonFano(mitadSuperior, codificacionActual + '0')
      generaShannonFano(mitadInferior, codificacionActual + '1')
   }
}
```

Una vez generados los códigos, se almacenaron en el archivo comprimido "*ShannonFano.Fan*". Y se almacenó el diccionario de codificación de dicho algoritmo, junto con los resultados de los cálculos en el archivo de texto "*ShannonFano.txt*".

4.1.3 Resultados

Luego de realizadas las codificaciones y generado posteriormente los archivos correspondientes a cada algoritmo, se los comparó en aspectos como:

- **La tasa de compresión**
- **El rendimiento**
- **La redundancia**

La **tasa de compresión** es el cociente entre el tamaño original del archivo sin comprimir y del archivo comprimido. Se expresa de la siguiente manera N : 1, siendo:

$$N = Tamaño\ archivo\ sin\ comprimir\ /\ Tamaño\ archivo\ comprimido$$

El **rendimiento** o eficiencia de un código es la relación entre el producto o el resultado obtenido y los medios utilizados. Para un código r-ario se utiliza la expresión:

$$\eta = \frac{H_r(S)}{L}$$

Siendo: $\mathbf{H_r(S)}$ = la entropía de la fuente
L = la longitud media del código

La **redundancia** es la reiteración de información incluida en textos o mensajes, que permite, pese a la ausencia de esta, rearmar su contenido. Una mayor redundancia implica menor información. Para un código r=ario se utiliza la expresión:

$$1 - \eta = \frac{L - H_r(S)}{L}$$

A continuación se recopilan los resultados obtenidos:

Algoritmo	Entropía	Longitud Media	Rendimiento	Redundancia	Tamaño Original	Tamaño Comprimido	Tasa de Compresión
Huffman	9,312	9.340	0.997	0.003	89.547 bytes	**161.257 bytes**	**0.5 : 1**
Shannon-Fano		9.394	0.992	0.008		**161.921 bytes**	**0.5 : 1**

Principalmente se observa que el resultado de la compresión no fue el esperado, ya que el tamaño de los archivos comprimidos es muy superior al del archivo original. Esto será analizado en detalle en la siguiente sección de este informe.

Se observa que por una mínima diferencia, el código generado por el algoritmo de Huffman es el más “óptimo”, ya que es el que presenta una menor longitud media que provocará que el tamaño del archivo comprimido sea de menor tamaño, si bien esta diferencia no es significativa la misma tiene una explicación lógica.

Los códigos que se generan por Shannon-Fano no son los óptimos en el sentido de que no poseen la menor longitud esperada posible, como es el caso de la codificación de Huffman. Este algoritmo logra que los símbolos con una mayor frecuencia ocupen el menor espacio posible y los de menor frecuencia, sean los que más espacio ocupen. Es por esto, que la codificación de Huffman posee una menor longitud media que la de Shannon-Fano.

En ambos métodos se observa un gran rendimiento, rozando incluso el valor límite 1, esto se debe a que se trata de métodos de compresión sin pérdida, es decir, los datos descomprimidos son prácticamente iguales a los datos originales. Pero es muy poco probable que la longitud media de los códigos alcance el valor óptimo de H(S), ya que la unidad mínima que acepta una computadora es el “**bit**”. No se admiten las “**fracciones de bits**”, por lo que al menos se necesita un bit por símbolo.

4.1.4 Archivos Comprimidos

Con respecto a los archivos comprimidos, estos se crean con el objetivo de ahorrar espacio a la hora de almacenarlos o transportarlos. Pero al momento de comprimirlos, es necesario un mecanismo que permita al receptor del archivo, poder descomprimirlo apropiadamente.

El mecanismo que se utilizó es el de incluir en el archivo comprimido, el diccionario de codificación, pero con la particularidad de que los campos de “símbolo” y “codificación” sean de longitud fija. La longitud fija del diccionario de codificación se corresponde con la longitud máxima de los símbolos y la longitud máxima de las codificaciones respectivamente. Esto se ve reflejado en los archivos “*Huffman.Huf*” y “*ShannonFano.Fan*”.

Es por esto que, si bien las compresion de los algoritmos por sí solas poseen un tamaño de 35.805 bytes (Huffman) y 36.469 bytes (Shannon-Fano), el hecho de incluir el diccionario dentro del archivo condiciona que el tamaño de los mismos sea de 161.257 bytes y 161.921 bytes, respectivamente.

Esto evidencia una contradicción con respecto a la compresión, ya que el resultado de esta es el opuesto al esperado. El responsable de esta contradicción es la técnica de compresión utilizada, ya que como se ha observado anteriormente, los métodos de compresión utilizados obtuvieron una gran eficiencia en sus codificaciones. Y dichas codificaciones representan poco más del 20% del tamaño del archivo en ambos casos.

Finalmente, para comprobar que la técnica de compresión utilizada fuese correcta, se reconstruyó el texto original a partir de los archivos comprimidos. Almacenando los resultados en los archivos “*reconstruccion.Huf*” y “*reconstruccion.Fan*”.

4.2 Segunda Parte

La descripción del canal se hace de forma matricial disponiendo las probabilidades condicionales:

Teniendo en cuenta las matrices calculadas (mostradas en el anexo) y las probabilidades de los símbolos de entrada se calcularon las diferentes probabilidades:

- **Probabilidad de salida**
- **Probabilidad “a-posteriori”**
- **Probabilidad del suceso simultáneo**

Y además las entropías

- **Entropía “a-priori”**
- **Entropía de salida**
- **Entropía “a-posteriori”**

(Nota : dichos valores de las matrices se encuentran en la sección anexo)

A partir de dichos datos podemos calcular también:

- **Equivocación**
- **Información mutua**
- **Entropía afín**
- **Pérdida**

Dichos datos se presentan a continuación:

Nro de Canal	Entropía A Priori	Entropía De Salida	Equivocación Ruido	Información Mutua	Entropía Afín	Pérdida
	H (A)	H (B)	H (A/B)	I (A,B)	H(A,B)	H(B/A)
Canal 1	2,170	1,579	2,153	0,018	3,731	1,561
Canal 2	1,948	1,987	1,922	0,027	3,909	1,961
Canal 3	2,547	1,973	2,502	0,025	4,476	1,948

Entropías a-posteriori

- ➔ **Canal 1** –> H(S / b(1)) = 2.202 , H(S / b(2)) = 2.170 , H(S / b(3)) = 2.097
- ➔ **Canal 2** –> H(S / b(1)) = 1.932 , H(S / b(2)) = 1.926 , H(S / b(3)) = 1.984 , H(S / b(4)) = 1.864
- ➔ **Canal 3** –> H(S / b(1)) = 2.500 , H(S / b(2)) = 2.488 , H(S / b(3)) = 2.543 , H(S / b(4)) = 2.486

Probabilidades de Salida

- ➔ **Canal 1** –> (0.297 , 0.331 , 0.372)
- ➔ **Canal 2** –> (0.256 , 0.260 , 0.199 , 0.286)
- ➔ **Canal 3** –> (0.200 , 0.247 , 0.224 , 0.330)

Con los valores expresados anteriormente, se llegó a las siguientes conclusiones:

A primera vista se puede observar que algunas de las matrices no son cuadradas por lo cual, ya sabemos que ese canal tendrá alguna clase de distorsión.

Además, el hecho de que las probabilidades del suceso simultáneo son todas distintas de cero nos da el indicio que es un canal con ruido y/o pérdida, pues dada una entrada todas las salidas son posibles o viceversa.

Para estos canales, la equivocación, la cual también se puede definir como ruido, o el número mínimo de preguntas binarias en promedio para determinar la entrada conociendo la salida, es en todos los casos, menor que la entropía a-priori.

Como **H(A) > H(A/B)** entonces en promedio nunca se pierde información al conocer la salida. Sin embargo, la diferencia de estos no es significativa lo que nos dice que en promedio, al conocer la salida, el aporte de información no será demasiado relevante.

Por otro lado la entropía a la salida es, solo en el canal 2 ligeramente mayor que la entropía a-priori, siendo en los demás canales menor que ésta.

Luego se comparó la entropía a-posteriori de cada símbolo, o sea, cuanto de la información de entrada sale del canal según la salida que se produzca, con la entropía a-priori (información media que ingresa al canal). Para el canal 1, solo con el primer símbolo hay más incertidumbre al observar la salida, ya que en este caso es mayor a la entrada. Para el canal 2, hay más incertidumbre al observar la salida para el símbolo 3. Y por último, para el canal 3 no hay casos en los cuales haya más incertidumbre a la salida.

En cuanto a la cantidad de información sobre la entrada que atraviesa el canal, o sea, la información mutua se puede decir que, como en todos los casos ésta es mayor a cero, no se pierde información en absoluto por el hecho de observar la salida del canal.

Además, se corroboró la simetría de la “Información Mutua” respecto de las variables ‘ai’ y ‘bj’, calculándola de ambos modos.

En cuanto a la pérdida, la cual se puede definir como el mínimo número de preguntas binarias en promedio para determinar la salida conociendo la entrada, el canal 1 es el que tiene una “pérdida” menor, mientras que los demás canales poseen un valor muy similar entre ellos, pero mayor al del primero.

Por último, observando la entropía afín y sabiendo que representa la incertidumbre del suceso simultáneo. Se puede decir que en todos los casos, como ésta es mayor que la entropía “a-priori”, la incertidumbre aumenta luego de pasar por el canal.

5. Conclusiones

Con respecto a la primera parte del trabajo, es notable la eficiencia de los algoritmos planteados por Huffman y Shannon Fano para la compresión del archivo a la hora de reducir el tamaño del mismo.

Pudimos evidenciar que el algoritmo de Huffman logra longitudes medias de palabra menores por lo que a la hora de realizar la compresión será más eficiente, si bien en el ejemplo trabajado la diferencia no es notoria, la misma está presente. Podemos atribuir a que Shannon Fano busca que todas las longitudes de palabras de código están a un bit de su ideal teórico, a diferencia de Huffman que busca la mínima longitud media de la palabra. Es por este motivo que en la actualidad el algoritmo de Shannon Fano ha pasado a un segundo plano y el mismo no es tan utilizado como el de Huffman.

También pudimos observar que, en general, los códigos generados por estos algoritmos no alcanzan el máximo rendimiento posible debido a la limitación que posee la unidad mínima de información que es el ‘bit’. Esto provoca que al utilizar este tipo de algoritmos, la performance del código mejore notoriamente, a expensas de un aumento exponencial del tamaño del diccionario de codificación.

Otra limitación recae en el mecanismo de compresión, dado que llegado el momento, tarde o temprano, estos archivos deben descomprimirse. Y para que esto sea posible es indispensable que dicho mecanismo sea conocido, tanto por el emisor del mensaje como su receptor. Y el costo negativo de esto es la reducción de la tasa de compresión.

Este costo negativo, para el caso de estudio de este informe, deja obsoleta a las codificaciones realizadas, ya que el hecho de tener que incluir el diccionario de codificación dentro de la compresión resulta en un archivo comprimido con un tamaño superior al original. Esto nos lleva a concluir que, utilizando este mismo mecanismo de compresión, se podrían llegar a analizar las siguientes situaciones para que la eficiencia de la compresión sea realmente eficiente:

- Incluir el diccionario de codificación dentro del archivo pero comprimido.
- Utilizar este mecanismo en archivos con mayor tamaño que el del caso de estudio.
- Siempre tener en cuenta que el diccionario de codificación estará ligado al archivo original, por lo que el tamaño de este puede variar entre archivos.

Por último, para que la compresión hubiese resultado eficiente utilizando este mecanismo y el mismo diccionario de codificación, el archivo original debió haber sido, aproximadamente, superior a 220.000 bytes.

Con respecto a la segunda parte; del estudio de los diferentes canales de información podemos resumir las siguientes conclusiones:

- Todos los canales tienen distorsión, dada por el ruido y/o pérdida.
- En promedio nunca se pierde información al conocer la salida.
- No se pierde información en absoluto por el hecho de observar la salida del canal.
- En todos los casos la incertidumbre aumenta luego de pasar por el canal.

6. Anexo

6.1 Canal 1

Matriz del Canal =
$$\begin{pmatrix} 0.30 & 0.27 & 0.43 \\ 0.36 & 0.40 & 0.24 \\ 0.30 & 0.27 & 0.43 \\ 0.27 & 0.40 & 0.33 \\ 0.30 & 0.36 & 0.34 \end{pmatrix}$$

Probabilidades a posteriori =
$$\begin{pmatrix} 0.202 & 0.121 & 0.303 & 0.272 & 0.101 \\ 0.163 & 0.121 & 0.244 & 0.362 & 0.108 \\ 0.231 & 0.065 & 0.347 & 0.266 & 0.091 \end{pmatrix}$$

Probabilidad del suceso simultáneo =
$$\begin{pmatrix} 0.060 & 0.054 & 0.086 \\ 0.036 & 0.040 & 0.024 \\ 0.090 & 0.081 & 0.129 \\ 0.081 & 0.120 & 0.099 \\ 0.030 & 0.036 & 0.034 \end{pmatrix}$$

6.2 Canal 2

Matriz del Canal =
$$\begin{pmatrix} 0,20 & 0,27 & 0,18 & 0,35 \\ 0,27 & 0,30 & 0,18 & 0,25 \\ 0,27 & 0,18 & 0,20 & 0,35 \\ 0,27 & 0,30 & 0,27 & 0,16 \end{pmatrix}$$

Probabilidades a posteriori =
$$\begin{pmatrix} 0,198 & 0,352 & 0,289 & 0,160 \\ 0,260 & 0,381 & 0,187 & 0,173 \\ 0,227 & 0,299 & 0,271 & 0,204 \\ 0,303 & 0,286 & 0,328 & 0,083 \end{pmatrix}$$

Probabilidad del suceso simultáneo =
$$\begin{pmatrix} 0,050 & 0,067 & 0,045 & 0,087 \\ 0,089 & 0,099 & 0,060 & 0,083 \\ 0,073 & 0,049 & 0,054 & 0,095 \\ 0,040 & 0,045 & 0,040 & 0,024 \end{pmatrix}$$

6.3 Canal 3

Matriz del Canal =
$$\begin{pmatrix} 0,20 & 0,27 & 0,18 & 0,35 \\ 0,27 & 0,27 & 0,30 & 0,16 \\ 0,18 & 0,20 & 0,27 & 0,35 \\ 0,27 & 0,30 & 0,18 & 0,25 \\ 0,20 & 0,27 & 0,27 & 0,26 \\ 0,18 & 0,27 & 0,30 & 0,25 \end{pmatrix}$$

Probabilidades a posteriori =
$$\begin{pmatrix} 0,138 & 0,124 & 0,166 & 0,310 & 0,129 & 0,133 \\ 0,154 & 0,102 & 0,132 & 0,285 & 0,143 & 0,164 \\ 0,112 & 0,124 & 0,223 & 0,186 & 0,156 & 0,189 \\ 0,189 & 0,058 & 0,252 & 0,225 & 0,131 & 0,144 \end{pmatrix}$$

Probabilidad del suceso simultáneo =
$$\begin{pmatrix} 0,030 & 0,041 & 0,027 & 0,053 \\ 0,027 & 0,027 & 0,030 & 0,016 \\ 0,036 & 0,040 & 0,054 & 0,070 \\ 0,068 & 0,078 & 0,045 & 0,063 \\ 0,028 & 0,038 & 0,038 & 0,063 \\ 0,029 & 0,043 & 0,048 & 0,040 \end{pmatrix}$$