Teoría de la Información
Trabajo Práctico Integrador N°2

Priano Sobenko Bautista
Legajo 15669
bautipriano@gmail.com
Rojas Agustin
Legajo 15757
agusaar@gmail.com
Matarazzo Tomas
tmatarazzo03@gmail.com

Fecha de Re-Entrega: 29/11/2022

Github: https://github.com/TomasMatarazzo/TeoriaDeLaInformacion-Facultad

Contenidos

Resumen

El trabajo consiste en leer un archivo de texto plano y poder comprimir su información por medio de dos métodos sin pérdida o reversibles conocidos y trabajados a lo largo de la cursada: Los algoritmos de Huffman y Shannon-Fano. Además, se realizará un análisis e interpretación del resultado final obtenido.

En la segunda parte del trabajo, se reciben tres canales, determinados su alfabeto de entrada y las probabilidades de emisión de cada símbolo de este, un alfabeto de salida y el conjunto de probabilidades condicionales de aparición de estos. Estas últimas describen la probabilidad de que se emita un símbolo de salida Bj habiendo ingresado un símbolo Ai. Con esta información, se busca calcular y analizar la equivocación, la información mutua y las propiedades de cada canal.

Introducción

La compresión de datos, es el proceso de codificación o conversión de datos de tal manera que consume menos espacio de memoria. Su aplicación reduce la cantidad de recursos necesarios para almacenar y transmitir datos.

Se puede hacer de dos maneras: compresión sin pérdida y compresión con pérdida. La compresión con pérdida reduce el tamaño de los datos al eliminar información innecesaria, mientras que no hay pérdida de datos en la compresión sin pérdida. En este informe estudio técnicas de compresión de datos sin pérdida. Se comprimirá un archivo de texto mediante las técnicas de compresión de Huffman y Shannon-Fano para poder analizar e interpretar los resultados obtenidos.

Respecto a la segunda parte, un canal de información es el medio por el que se transmite la información desde una fuente de información a su destino.

Estos se estudian ya que se pone en manifiesto dos situaciones. La primera, es que la fuente emisora y el receptor pueden trabajar con símbolos de distinta naturaleza. La segunda, es que el canal lógicamente puede contener ruido, generando cambios aleatorios e incontrolables en los símbolos transmitidos, produciendo errores en la transmisión de la información. Se estudiarán herramientas matemáticas como la equivocación, distintas entropías y la información mutua para evaluar qué tan exitosa fue la transmisión en un canal discreto y sin memoria.

Estas se herramientas se aplican sobre su alfabeto de entrada con la probabilidad de emisión de sus símbolos, su alfabeto de salida y sobre su matriz de matriz de probabilidades condicionales.

Desarrollo

Primera parte

El objetivo de esta primera parte fue a partir de los distintos métodos estudiados en la materia, comprimir el archivo dispuesto por la cátedra, es un archivo de texto el cual consta de 60 caracteres distintos y aproximadamente 4000 palabras distintas.
Se comprime utilizando los algoritmos de compresión Huffman y Shannon-Fano, a partir de los resultados obtenidos se compara con el archivo sin comprimir para poder sacar deducciones respecto a las tasas de compresión, rendimiento y redundancia.
Definiciones a tener en cuenta para las compresiones realizadas por los algoritmo de ***Huffman*** y ***Shannon-Fano***

- **Entropía de la fuente:** La probabilidad de que aparezca el símbolo si es precisamente P(si), de modo que la cantidad media de información por símbolo de la fuente es

$$H = \sum_{s}^{i} P(Si) * (-log\ P(Si))$$

- **Longitud media:**
    - Sea un código bloque que asocia los símbolos de una fuente S = { s1 , s2 , ..., sq } con las palabras X1 , X2 , ..., Xq .
    - Las probabilidades de los símbolos de la fuente son Pl , P2 , ..., Pq
    - Las longitudes de las palabras l1 , l2 , ..., lq.

$$L=\sum_{i=1}^{q} p_i l_i$$

- **Rendimiento y Redundancia:**

  Es una medida que cuantifica la cantidad de información que otorga un código. La medida para saber cuánta información es redundante se la denomina redundancia.

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

- **Tasa de compresión:**

  Medida para cuantificar cuánto se comprime el archivo original en relación a**l** archivo comprimido

$$Tasa\ de\ Compresión\ =\ Tamaño\ archivo\ original\ /\ Tamaño\ archivo\ comprimido$$

Algoritmo de Huffman

El algoritmo de Huffman es una técnica de compresión de datos sin pérdida. La idea de este algoritmo se basa en el uso de la “codificación de longitud variable”, se identifican aquellas palabras que aparecen con menor frecuencia dentro del archivo y se les asignan representaciones, palabras códigos, de mayor extensión.

```
Proceso algoritmoHuffman()
  //frecPalabra: mapeo de [palabra, frecuencia]
  pQ: ColaPrioridad
  para cada pal en frecPalabra:
    CrearNodoHoja(nodoHoja)
    nodoHoja.pal = pal
    nodoHoja.frec = pal.frecuencia
    Inserta(pQ , nodoHoja)

  mientras (tamaño pQ > 1):
    //El nodo resultante sera una fusion de las dos nodos con menor frecuencia
    CrearNodo(nodo)
    nodo.hijoIzq = Elimina(pQ)
    nodo.hijoDer = Elimina(pQ)
    nodo.frec = nodo.HijoIzq.frec + nodo.HijoDer.frec
    Inserta(pQ, nodo)

  //Etiqueto el arbol concantenando "1s" y "0s" de forma recursiva
  generaCodigoHuffman(raiz = Elimina(pQ) , "")
```

Luego de realizada esta técnica se obtuvieron los siguientes resultados:

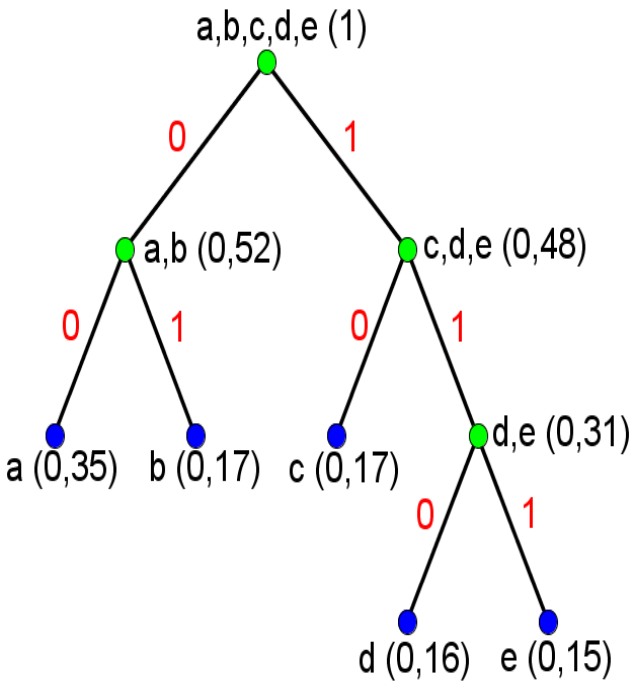
Algoritmo De Huffman	Entropía	Longitud Media	Rendimiento	Redundancia	Tamaño original (bytes)	Tamaño comprimido (bytes)	Tasa de compresión
DatosTP2.txt	9,228	9,256	0,997	0,003	85.881	255.286	0,3364

Algoritmo de Shannon-Fano

El algoritmo de Shannon Fano es una técnica de codificación de entropía para la compresión de datos multimedia sin pérdidas. Es un procedimiento subóptimo para construir un código, que alcanza una cota de $L \leq H(S) + 2$,se basa en asignar un código a cada símbolo en función de sus probabilidades de ocurrencia. Para el desarrollo del algoritmo se realizan los siguientes pasos.

- ➔ Se crea una lista de frecuencias para el conjunto de símbolos dados, de modo que se conozca la frecuencia relativa de aparición de cada símbolo.
- ➔ Se ordena la lista de símbolos en orden decreciente de probabilidad, los más probables a la izquierda y los menos probables a la derecha.
- ➔ Divida la lista en dos partes, con la probabilidad total de que la parte izquierda y derecha sean las más cercanas posibles..
- ➔ Asigne el valor 0 a la parte izquierda y 1 a la parte derecha.
- ➔ Repita los pasos 3 y 4 para cada parte hasta que todos los símbolos se dividan en subgrupos individuales.

A partir de estos pasos se planteó el siguiente pseudocódigo para la implementación del algoritmo.



El objetivo del algoritmo es formar un árbol como el que se encuentra en la imagen, los caracteres representan los símbolos que van a tener sus respectivas probabilidades, según su probabilidad se asigna si va al nodo derecho o izquierdo. A la nueva rama izquierda se le asigna el valor de 0 y a la derecha de uno. Así para cada nodo hasta que solamente quede únicamente con un único valor.

Una vez se tiene el árbol formado es necesario obtener los códigos para cada símbolo, cada símbolo es una hoja del árbol binario. Entonces se recorre el árbol verificando en qué nodos se encuentra la palabra hasta llegar al nodo hoja.El recorrido realizado es el código para esa palabra. Por ej, para el símbolo d el código es 110, para el b es 01.

```
function creaArbol( Nodo nodo){

  if (nodo.simbolos.size != 1){
    parteIzq = lista vacia
    parteDer = lista vacia
    indiceDondeDivido = metodoParaObtenerIndice(nodo.simbolos)
    for (nodo.simbolos.size()){
        if (i < indiceDondeDivido)
            parteIzq.add( nodo.simbolos.get(i))
        else
            parteIzq.der( nodo.simbolos.get(i))
    }
    agregoHijos(nodo , parteIzq,parteDer)
    crearArbol( nodo.getParteIzq())
    crearArbol( nodo.getParteDer())
  }
}
```

La función *crearArbol* es una función recursiva la cual tiene como parámetro un Nodo del árbol. Como primer paso se verifica que no sea un nodo hoja (es decir que el nodo solamente contenga un símbolo) si no lo es , a partir de los símbolos del nodo se obtiene el índice que divide a la mitad según la probabilidad total de la parte izq y der. Se prosigue a entrar en un ciclo y agregó a una lista los símbolos correspondientes a su respectivo lado. Finalmente, creo el nodo izquierdo y nodo derecho con las listas y hago la recursión para ambos nodos.

```
function obtenerCodigo( String simbolo , Nodo nodo, String[] codigo , int i ){
  if (nodo.getSimbolos.size != 1){
    boolean estaSimbolo = nodo.getParteIzq.getSimbolos.contains(simbolo)
    if (estaSimbolo){
      codigo[i] += '0';
      obtenerCodigo(simbolo , nodo.getParteIzq , codigo,i)
    }
    else {
      codigo[i] += '1';
      obtenerCodigo(simbolo , nodo.getParteDer , codigo,i)
    }
  }
}
```

La función *obtenerCodigo* se utiliza para obtener el camino dependiendo el símbolo que se pase como parámetro. Al igual que la función anterior verificamos que no sea un nodo hoja, si no lo es usamos una variable booleana llamada está que verifica si el símbolo se encuentra en el lado izquierdo, si es así hacemos la recursión por ese lado de no serlo realizamos la recursión para el lado contrario.Siempre se modifica el código dependiendo el lado que se encuentre.

Una vez aplicada las funciones los resultados mediante el algoritmo fueron los siguientes:

Algoritmo De Shannon	Entropía	Longitud Media	Rendimiento	Redundancia	Tamaño original (bytes)	Tamaño comprimido (bytes)	Tasa de compresión
DatosTP2.txt	9,228	9,307	0,992	0,008	85.881	255.380	0,3362

Comparación desde el punto de vista de la compresión, el rendimiento y la redundancia.

Algoritmo De Huffman	Rendimiento	Redundancia	Tasa de compresión
DatosTP2.txt	0.997	0,003	0,3364

Algoritmo De Shannon-Fano	Rendimiento	Redundancia	Tasa de compresión
DatosTP2.txt	0.992	0,008	0,3362

Análisis e interpretación de resultados obtenidos

Luego de la compresión del archivo aplicando los algoritmos de *Shannon-Fano* y *Huffman* se observa que la compresión generada no es óptima. La carga de la tabla de decodificación en el archivo provoca que el archivo final comprimido sea de mucho mayor tamaño que el original. La codificación de las palabras código y de las palabras fuente en forma estática generan ineficiencias en la compresión, desaprovechando espacio.

La tabla de decodificación cargada dentro del binario se encuentra compuesta de la siguiente manera:

Cantidad de palabras de la fuente	longMaxPalabraCodigo	longMaxPalabraFuente	Palabra Codigo	Palabra Fuente
4 bytes	4 bytes	4 bytes	2 bytes * longMaxPalabraCodigo	2 bytes * longMaxPalabraFuente

Cantidad de palabras de la fuente:

Este bloque se utilizará para generar el diccionario (Palabra Codigo, PalabraFuente) que permitirá descomprimir el código. Con él se conocerá el número de palabras únicas que tiene el código y por lo tanto la extensión del diccionario.
(Bloque único dentro del binario)

longMaxPalabraCodigo

Se utiliza para almacenar en el binario, en forma estática, la palabra código. Esta longitud máxima de palabra código variará dependiendo del algoritmo de compresión.
(Bloque único dentro del binario)

longMaxPalabraFuente

Se utiliza para almacenar en el binario, en forma estática, la palabra fuente. Esta longitud máxima de palabra código dependerá de las palabras que conforman el texto a comprimir.
(Bloque único dentro del binario)

Palabra Código + Palabra Fuente:

Este bloque se repetirá en función de la cantidad de palabras de la fuente. Serán los elementos del diccionario. Los 2 bytes se utilizan debido a que un tipo de dato Caracter utiliza 2 bytes para su almacenamiento. Se utilizan las longitudes máximas para realizar una carga estática dentro del archivo.
(Bloque * Cantidad de palabras de la fuente)

La compresión del código, sin tener en cuenta el gran incremento del tamaño del archivo por la carga de la tabla de descompresión, demuestra que ambas técnicas han sido eficaces, Se observa además que el rendimiento se encuentra muy cercano a la eficiencia máxima del 100%, demostrando así, que todos los datos representarán información y que no existirá pérdida de datos.

De igual manera, como nos encontramos trabajando con métodos de compresión sin pérdida o reversibles se conoce que la longitud media del código se encuentra acotada por la entropía. Una vez más los resultados arrojados por los algoritmos de Huffman y Shannon demuestran esto, arrojando en forma aproximada una H <= L.

La compresión del código se aprecia de igual manera comparando la longitud máxima de las palabras código con la longitud máxima de las palabras fuente, siendo para ambas técnicas de 17 y 14 respectivamente. Sin embargo, esto no concluye en una compresión eficaz debido a que se tiene que contemplar la tabla de decodificación dentro del archivo y es con este incremento que la compresión no es exitosa. Probablemente para archivos de mayor extensión, la tabla, no sea un inconveniente en la compresión, pero para el archivo con el que

estamos trabajando si lo es. La tabla podría disminuir su tamaño si no fuera escrita en forma estática, esto se lograría agregando un bloque detrás de cada palabra fuente indicando su longitud correspondiente, esto provocaría que se lean únicamente los bytes necesarios para representar estas palabras y no se desperdicie espacio leyendo una cantidad de bytes fijos dependiendo de una longitud máxima.

Segunda Parte

Un canal es el medio de transmisión por el cual la fuente se comunica con el receptor del mensaje. La introducción del concepto de canal de información nos lleva a considerar la posibilidad de cometer errores durante el proceso de transmisión. Los símbolos de entrada se eligen de acuerdo con sus probabilidades P (a1), P (a2), . . ., P (an). Los símbolos de salida aparecerán de acuerdo con otro conjunto de probabilidades: P (b1), P (b2), ..., P (bm).

Se dice que un canal discreto no tiene memoria si la distribución de probabilidad de una salida depende sólo de la entrada correspondiente y es condicionalmente independiente de previas entradas o salidas del canal. En nuestro caso, los canales estudiados no poseen memoria, por lo tanto se puede definir como matrices de probabilidades condicionales donde cada matriz define P(bj/ai). Esto es es la probabilidad de recibir a la salida el símbolo bj, cuando se envía el símbolo de entrada ai. De esta forma, un canal de información queda completamente definido por su matriz.

A partir de estos parámetros, si se tienen X símbolos de entrada e Y símbolos de salida, se puede calcular la probabilidad de salida de un símbolo sin conocer el símbolo de entrada de la siguiente manera:

$$P(b_j) = \sum_{i=1}^{X} P(a_i) * P(b_j / a_i) \qquad \forall\, j = 1, \dots, Y$$

Este cálculo se desarrolló con la siguiente función,alojando las probabilidades todas juntas en un vector:

```
funcion calculaB(A,matrizP(bj/ai)){
B[] = 0;
for(i=0;i<CantSimbolosSalida;i++){
    for(j=0;j<CantSimbolosEntrada;j++){
        B[i] += matrizP(bj/ai)[j][i] * A[j];
    }
}
return B;
}
```

Además de P(bj), existe otro conjunto de probabilidades relativas a un canal que puede calcularse a partir de P(ai) y P(bj/ai):

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

Estas probabilidades condicionales, denominadas hacia atrás, definen la probabilidad que se haya ingresado el símbolo ai habiendo salido el símbolo bj. Además, se puede calcular el suceso simultáneo de (ai , bj) como:

$$P(ai\ , bj) = P(ai/bj) * P(bj) = P(bj/ai) * P(ai)$$

Así, quedan definidas las probabilidades “a priori” y “a posteriori”. En el primer caso se define la probabilidad que salga un elemento, tanto de salida como de entrada, sin conocer que ocurrió en el otro extremo del canal. Estas son P(A) y P(B). El hecho de conocer el símbolo que entra o sale en el canal hace que varíe la probabilidad con la se pueden observar los distintos símbolos en el otro extremo del canal. Así quedan definidas, P(ai/bj), o sea, la probabilidad de que haya ingresado el símbolo ai sabiendo que salió bj y P(bj/ai), la probabilidad de que salga bj sabiendo que ingresó al canal bj.

A el conjunto de cada una de estas probabilidades se le puede calcular la entropía y dependiendo si el conjunto es “a priori” o “posteriori”, esta será del mismo tipo.

- Entropía “a priori” de A:

$$H(A) = \sum_{i=1}^{X} P(ai) * \log\left(\frac{1}{P(ai)}\right)$$

- Entropía “a posteriori” de A, recibido bj:

$$H(A/bj) = \sum_{i=1}^{X} P(ai/bj) * \log\left(\frac{1}{P(ai/bj)}\right)$$

A partir del primer teorema de Shannon, se entiende que, H(A) es el número medio de binits necesarios para representar un símbolo de una fuente con una probabilidad a priori P (ai) i = 1, 2, … , X. Por el otro lado, H(A/bj) se comprende como el número medio de binits necesarios para representar un símbolo de una fuente con una probabilidad a posteriori P(ai/bj), i = 1, 2, … , X.

Con los conceptos descritos hasta el momento, podemos definir la entropía media “a posteriori” como:

$$H(A/B) = \sum_{i=1}^{Y} P(b_i) * \mathrm{H}(\mathrm{A}/b_i)$$

Esta también se define como la equivocación de A con respecto a B a través del canal, es una medida fundamental ya que evalúa:

- La información que queda en A después de observar B.
- La pérdida de información sobre A causada por el canal.
- La cantidad de información sobre A que no deja pasar el canal.

Se puede definir también H(B/A) que determina la cantidad media de preguntas binarias para determinar la salida conocida. Esta se denomina como pérdida del canal, H(A/B), en cambio, es el ruido del canal.

Su cálculo se desarrolló como:

```
funcion calculaEntropiaPosteriori(matrizP(ai/bj),simboloSalida)){
    res=0;
    for(i=0;i<CantSimbolosEntrada;i++){
        res += matrizP(ai/bj)[i][simboloSalida] * log2(1/matrizP(ai/bj)[i][simboloSalida]);
    }
}
return res;
}
```

Además, se puede calcular la entropía afín, la cual evalúa la incertidumbre del suceso simultáneo medio, se puede calcular de distintas formas:

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

$$H(A,B) = H(B) + H(A/B)$$

$$H(A,B) = H(A) + H(B/A)$$

Entonces, una vez que contamos con H(A), la cual nos define la cantidad media de binits para determinar un símbolo de entrada ai y con H(A/B) que nos define la cantidad media de binits necesarios para determinar un símbolo de entrada ai conociendo el símbolo de salida bj, podemos definir la información aportada por el símbolo bj. Esta se denomina información mutua y se define como:

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

Esta medida también se puede entender como la cantidad de información de A que atraviesa el canal o como la incertidumbre de A que desaparece al conocer la salida del canal. Por ende, nos indica que tan bien o mal se está usando un canal, así permite elegir el canal adecuado para la transmisión de símbolos generados por una fuente fija.

Antes de presentar los resultados y análisis cabe aclarar que la base de todos los logaritmos son en base 2, por lo tanto todos los resultados serán en bits.

Canal 1

Entropía “a priori” de la fuente: H(A) = 2.171 bits
Entropía “a priori” de la salida del canal: H(B) = 1.583 bits

Equivocación: H(A/B) = 2.154 bits
Información Mutua: I(A,B) = 0.017 bits

Entropía Afin: H(A,B) = H(B)+H(A/B) = 3.737 bits

Equivocación: H(B/A) = 1.566 bits
Información mutua: I(B,A) = 0.017 bits
Entropía Afin: H(A,B) = H(A)+H(B/A) = 3.737 bits

Entropía “a posteriori” recibido B1: H(A/b1) = 2.202 bits
Entropía “a posteriori” recibido B2: H(A/b2) = 2.185 bits
Entropía “a posteriori” recibido B3: H(A/b3) = 2.079 bits

La diferencia entre H(A) y H(A/B) no es grande, esto quiere decir que el aporte de información en promedio, que genera conocer la salida no es grande, esto se puede ver en la baja información mutua obtenida.

Por otro lado, se observa que el símbolo b1 no aporta información ya que genera más incertidumbre conocerlo que no hacerlo. El símbolo b3 es el único que aporta un conocimiento considerable para determinar el símbolo de entrada al conocerlo.

Canal 2

Entropía “a priori” de la fuente: H(A) = 1.948 bits
Entropía “a priori” de la salida del canal: H(B) = 1.993 bits

Equivocación: H(A/B) = 1.915 bits
Información Mutua: I(A,B) = 0.033 bits
Entropía Afin: H(A,B) = H(B)+H(A/B) = 3.908 bits

Equivocación: H(B/A) = 1.960 bits
Información mutua: I(B,A) = 0.033 bits
Entropía Afin: H(A,B) = H(A)+H(B/A) = 3.908 bits

Entropía “a posteriori” recibido B1: H(A/b1) = 1.925 bits
Entropía “a posteriori” recibido B2: H(A/b2) = 1.934 bits
Entropía “a posteriori” recibido B3: H(A/b3) = 1.985 bits
Entropía “a posteriori” recibido B4: H(A/b4) = 1.820 bits

En este caso la información aportada por la salida para conocer la entrada es algo mayor. De los 4 hay un solo símbolo que genera incertidumbre al conocerlo que es el b3, y el b4 aporta una gran cantidad de información en relación al resto. Esto causa que en promedio el aporte sea mayor, causando finalmente una suba en la información mutua.

Canal 3

Entropía “a priori” de la fuente: H(A) = 2.527 bits
Entropía “a priori” de la salida del canal: H(B) = 1.995 bits

Equivocación: H(A/B) = 2.496 bits
Información Mutua: I(A,B) = 0.031 bits
Entropía Afin: H(A,B) = H(B)+H(A/B) = 4.491 bits

Equivocación: H(B/A) = 1.963 bits
Información mutua: I(B,A) = 0.031 bits
Entropía Afin: H(A,B) = H(A)+H(B/A) = 4.491 bits

Entropía “a posteriori” recibido B1: H(A/b1) = 2.476 bits
Entropía “a posteriori” recibido B2: H(A/b2) = 2.526 bits
Entropía “a posteriori” recibido B3: H(A/b3) = 2.539 bits
Entropía “a posteriori” recibido B4: H(A/b4) = 2.429 bits

En este último caso la información mutua sigue siendo baja, el símbolo que más aporta información es el b4 y el b2 y b3 no aportan nada. Cabe aclarar nuevamente que la información mutua evalúa el aporte medio, por lo tanto se ve “arrastrada” por el aporte de b1 y b2.

Como conclusiones generales a los tres canales se puede afirmar que las informaciones mutuas fueron bajas, significando que los canales elegidos no fueron buenos para transmitir la información generada por las fuentes. Sin embargo, ningún alfabeto de salida fue independiente del de entrada(I(A,B)>0). Además, se pudo demostrar la propiedad de reciprocidad tanto de la entropía afín como de la información mutua.

Conclusión

En el trabajo planteado primero se logró comprimir un archivo de texto tanto por el algoritmo de Huffman como Shannon-Fano. Estas técnicas son de vital importancia en un mundo donde la información abunda y los archivos tanto de audio, video o incluso programas tienen tamaños cada vez más grandes. Sin embargo, el mayor objetivo cumplido es que se pudo realizar un profundo análisis de los resultados obtenidos, para entender qué tan nobles eran las compresiones por medio de herramientas provistas por la cátedra como el rendimiento, la redundancia y la tasa de compresión.

Por otro lado, se estudiaron y se caracterizaron tres canales con un alto nivel de detenimiento. Nuevamente, nos encontramos en un entorno donde el internet y los sistemas distribuidos son herramientas básicas del día a día, causando que la transmisión constante de información tenga que ser prácticamente perfecta. Esto genera que el estudio de canales sea de una gran importancia. Se pudo obtener características de cada canal para poder construir un análisis completo de él y determinar si el canal y la fuente fueron bien elegidos o no.

Para concluir, se pudieron aplicar todas las herramientas vistas en la cátedra y se pudo llegar a conclusiones que no hubieran sido posibles de obtener sin un profundo conocimiento de la teoría de la información. Por lo tanto, se concluye que el trabajo fue exitoso.