Teoría de la Información
Trabajo Integrador N°1

Grupo N°: 5

Integrantes:
Noelia Echeverría (noelia.echeverria@fi.mdp.edu.ar) Leg. 4087
María Camila Ezama (camilaezama2000@gmail.com) Leg. 15103
Juan Cruz Mateos (juanczmt@gmail.com). Leg. 15134

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

Índice

Resumen

Este trabajo está dividido en dos partes. En la primera parte se realizan distintos cálculos con el objetivo de caracterizar la fuente de información a partir de un archivo que contiene miles de unos y ceros. Se calcula la cantidad de información y la entropía de fuente, tomando palabras de código de distintas longitudes. A continuación, considerando un agrupamiento de a pares, se calculan las probabilidades condicionales para determinar si la fuente es de memoria nula o no. Por último se calcula el vector de probabilidades estacionarias y se analiza la ergodicidad de la fuente.

En la segunda parte, se determinan las características de los códigos utilizados previamente, y se comprueba si son unívocamente decodificables, automáticos y compactos. Para esto, se calculan sus longitudes medias y se verifica si cumplen con la inecuación de Kraft-MacMillan. En base a estos cálculos y a la entropía previamente calculada se verifica si los códigos son compactos, y se calcula la eficiencia y redundancia de los mismos.

1 - Introducción

Este trabajo está dividido en dos partes bien diferenciadas. La primera parte hace referencia a la información y sus fuentes, mientras que la segunda parte hace referencia a las propiedades de los códigos. En esta introducción se verá un pequeño resumen de los conceptos teóricos más relevantes para cada uno de estos dos grandes temas.

1.1 - La información y sus fuentes:

La fuente de información más sencilla que podemos imaginar, conocida con el nombre de fuente de memoria nula, es aquella que transmite una secuencia de símbolos estadísticamente independientes, los cuales pertenecen a un alfabeto fuente finito y fijo S= {S_1, S_2,...,S_q}. Esta fuente puede ser descrita completamente mediante el alfabeto fuente S y las probabilidades con que los símbolos se presentan: P(S_1), P(S_2), …, P(S_q).

Dado que la presencia del símbolo S_i corresponde a una cantidad de información igual a:

$$I(Si) = log(\frac{1}{P(Si)})$$

es posible obtener la información media suministrada por la fuente o *entropía* como:

$$H(s) = \sum_{i=1}^{q} P(Si).log(\frac{1}{P(Si)})$$

Si se agrupan las salidas de la fuente de memoria nula en paquetes de *n* símbolos, se tendrá una fuente de memoria nula de q^n símbolos {$\sigma_{1,}\sigma_2$, …, σ_{qn}}, conocida con el nombre de extensión de orden *n* de *S* o S^n. Si σ_i representa la secuencia {S_{i1}, S_{i2},..., S_{in}}, entonces la probabilidad asociada a σ_i será P(σ_i) = P_{i1} . P_{i2}. … P_{in} .

Dado que un símbolo de la extensión de orden *n*, S^n, de la fuente de memoria nula *S*, corresponde a *n* símbolos de *S,* es de esperar que la entropía por símbolo de S^n sea *n* veces mayor que la de *S. (Abramson N., Teoría de la Información y Codificación. 1981)*

$$H(S^n) = \sum_{i=1}^{qn} P(\sigma i).log(\frac{1}{P(\sigma i)}) = nH(s)$$

La fuente de memoria nula hasta aquí considerada resulta demasiado limitada en algunas aplicaciones. Un tipo de fuente de *q* símbolos más general, consiste en aquella en que la presencia de un determinado símbolo S_i depende de un número finito *m* de símbolos precedentes. Tal fuente es conocida con el nombre de *Fuente de Markov de orden m* y viene definida por el alfabeto S y el conjunto de probabilidades condicionales:

P(S_i/S_{i1}, S_{i2}, …, S_{im}) para *i*=1, 2, …, *q*; *iv* = 1, 2, …

En una fuente de Markov de orden *m*, la probabilidad de un símbolo cualquiera viene determinada por los *m* símbolos que lo preceden. Puesto que existen *q* símbolos distintos, una fuente de Markov de orden *m* admitirá q^m estados posibles.

Cuando una fuente de Markov, observada por un tiempo suficientemente largo, emite (con probabilidad 1) una secuencia típica de símbolos, se dice que la misma es ergódica.

Existe una distribución de probabilidades única para un conjunto de estados de una fuente de Markov ergódica, y los estados en cualquier secuencia suficientemente larga, se presentarán (con probabilidad 1) de acuerdo con esta distribución. Esta distribución única recibe el nombre de *distribución estacionaria del proceso ergódico de Markov*. La distribución estacionaria no depende de la distribución inicial con que los estados son escogidos y puede calcularse directamente a partir de las probabilidades condicionales de los símbolos.

Por último, se debe mencionar que una fuente de memoria nula se puede considerar un caso muy especial de una cadena de Markov, con *m=0*. La dependencia de *m=0* símbolos precedentes denota la independencia entre los símbolos.

1.2 - Propiedades de los códigos

Siendo S = $\{S_1, S_2 \ldots S_q\}$ el conjunto de símbolos de un alfabeto dado, se define un código como la correspondencia de todas las secuencias posibles de símbolos de S a secuencias de símbolos de algún otro alfabeto X =$\{x_1, x_2 \ldots x_r\}$. S recibe el nombre de alfabeto fuente y X alfabeto código.

Partiendo de esta definición, es posible distinguir distintas clasificaciones de códigos. Un código bloque es aquel que asigna cada uno de los símbolos del alfabeto fuente S a una secuencia fija de símbolos del alfabeto código X. Esas secuencias fijas (secuencias de x,) reciben el nombre de palabras código. Este se denomina *no singular* si todas sus palabras son distintas y *unívocamente decodificable* si, y solamente si, su extensión de orden n definida anteriormente, es no singular para cualquier valor finito de n. Además, será instantáneo si ninguna palabra del código coincide con el prefijo de otra. Se denomina prefijo de una palabra $X_i = x_1^i, x_2^i \ldots x_m^i$ a la secuencia de símbolos $(x_1^i, x_2^i \ldots x_j^i)$ donde j ≤ m.

Para distinguir estas características se cuenta con las inecuaciones de Kraft y McMillan. A partir de la inecuación de Kraft se tiene que la condición necesaria y suficiente para la existencia de un código instantáneo de longitudes $l_1, l_2 \ldots l_q$ es que

$$\sum_{i=1}^{q} 2^{-l_i} \leq 1$$ para un alfabeto código de 2 símbolos.

McMillan aprovechó la inecuación de Kraft y demostró que si existe un código unívoco entonces se cumple:

$$\sum_{i=1}^{q} 2^{-l_i} \leq 1$$

1.3 - Codificación de fuentes de información

Sea un código bloque que asocia los símbolos de una fuente $S_1, S_2 \ldots S_q$, con las palabras $x_1, x_2 \ldots x_q$. Suponiendo que las probabilidades de los símbolos de la fuente son $P_1, P_2 \ldots P_q$ y las longitudes de las palabras $l_1, l_2 \ldots l_q$,. se define la longitud media del código, L como:

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

Considerando un código unívoco que asocia los símbolos de una fuente S con palabras formadas por símbolos de un alfabeto r-ario, éste será compacto (respecto a S) si su longitud media es igual o menor que la longitud media de todos los códigos unívocos que pueden aplicarse a la misma fuente y el mismo alfabeto.

Dada una fuente S, de símbolos $S_1, S_2 .... S_q$ y probabilidades $P_1, P_2 .... P_q$, es posible obtener un código instantáneo óptimo aplicando el algoritmo de Huffman, o bien códigos subóptimos aplicando el algoritmo de Shannon-Fano, los cuales no siempre alcanzan la menor longitud esperada posible. Se definen el rendimiento y la redundancia de un código como $\eta = \frac{H_r(S)}{L}$ y $1 - \eta = \frac{L - H_r(S)}{L}$ respectivamente.

2 - Desarrollo

2.1 - Primera parte

Dado un archivo anexo1-grupo5.txt el cual está formado por miles de unos y ceros y considerando los siguientes 3 escenarios:

- Escenario 1: Palabras código de 5 dígitos
- Escenario 2: Palabras código de 7 dígitos
- Escenario 3: Palabras código de 9 dígitos

2.1.1 - Inciso a

Se calcularon la cantidad de información y la entropía de la fuente. Los pseudocódigos utilizados para su cálculo se detallan a continuación.

Pseudocódigo para la lectura del archivo y cálculo de frecuencias de símbolos:

Leer del archivo palabras de x dígitos y contar la frecuencia de sus ocurrencias utilizando una variable de tipo diccionario.

```
x = cantidad de dígitos escenario
frecuencias := diccionario(string, int)
abrir archivo anexo.txt
total = 0
mientras no fin de archivo:
        palabra = leer x caracteres
        frecuencias[palabra] += 1
        total += 1
cerrar archivo
```

Pseudocódigo para el cálculo de probabilidades de cada símbolo:

Dividir cada frecuencia obtenida por el total de símbolos leídos del archivo.

```
probabilidades := diccionario(string, double)
para cada clave en frecuencias:
        probabilidades[clave] = frecuencias[clave] / total
```

Pseudocódigo para el cálculo de la cantidad de información de cada símbolo:

Para cada evento calcular la cantidad de información asociada utilizando la probabilidad obtenida anteriormente.

```
cantidad_información := diccionario(string, double)
para cada clave en frecuencias:
	cantidad_información[clave] = cant_info(probabilidades[clave])

función cant_info(probabilidad):
	return -1.0 * log2(probabilidad)
```

Pseudocódigo para el cálculo de la entropía de la fuente:

Conociendo la probabilidad y la cantidad de información de cada evento calcular la entropía de la fuente.

```
	entropía  = 0
	para cada clave en frecuencias:
		entropía += probabilidad[clave] * cantidad_información[clave]
```

Resultados obtenidos

En la Tabla I se muestran los resultados de las entropías calculadas para cada uno de los tres escenarios. La cantidad de información de información para cada escenario se puede consultar en los archivos anexos *resultados5digitos.txt*, *resultados7digitos.txt* y *resultados9digitos.txt*.

	Escenario 1	Escenario 2	Escenario 3
Entropía H(S) [bits]	4.996012045108118	6.9790094546532	8.895015764498266

Tabla I: Entropías calculadas para cada uno de los escenarios.

Discusión y conclusiones

La entropía se define como la cantidad de bits necesarios para representar la información, siendo siempre un valor positivo y tomando su valor máximo $H_{max}(S) = \log_2 n$ cuando los *n* símbolos son equiprobables.

Como en los casos bajo estudio la cantidad de símbolos *n* coincide con las 2^k combinaciones posibles (con k=5, 7 y 9), y siendo las mismas casi equiprobables, es esperable que, por lo tanto, la entropía se aproxime al valor de *k*. Se puede consultar la tabla I para verificar esta aseveración y los archivos anexos para visualizar las probabilidades de cada evento para cada escenario.

2.1.2 - Inciso b

Posteriormente se consideró la fuente como una secuencia de símbolos (cuatro) codificados como 00, 01, 10, 11 y bajo esta suposición se calcularon las probabilidades condicionales. El pseudocódigo utilizado para su cálculo se detalla a continuación.

Pseudocódigo para el cálculo de probabilidades condicionales:

Leer del archivo un símbolo (anterior), leer el siguiente (actual) y acumular la frecuencia de ocurrencia del actual dado que se leyó el anterior en una matriz y la frecuencia de ocurrencia del actual. Dividir cada columna j de la matriz por el valor acumulado de frecuencia j.

```
# lectura del archivo y cálculo de frecuencias
probabilidades := matriz[4][4]
```

```
frecuencias := vector[4]
abrir archivo anexo.txt
anterior = leer primer símbolo en archivo
mientras no fin de archivo:
	actual = leer símbolo de archivo
	probabilidades[actual][anterior] += 1
	frecuencias[anterior] += 1
	anterior = actual
cerrar archivo

# cálculo de probabilidades condicionales
para cada columna de probabilidades:
	para cada fila de probabilidades:
		probabilidades[fila][columna] /= frecuencias[columna]
```

Resultados obtenidos

En la tabla II se muestran los resultados de las probabilidades condicionales obtenidas (matriz de transición de estados).

Matriz Transición	00	01	10	11
00	0.259360277708902	0.264654527313545	0.259482082134449	0.240473738414006
01	0.255145053310191	0.254676976802195	0.248757520272038	0.259268795056643
10	0.246466650136375	0.243701671239711	0.241433429244049	0.23918640576725
11	0.239028018844533	0.23696682464455	0.250326968349464	0.261071060762101

Tabla II: Matriz de transición de estados considerando la codificación 00, 01, 10, 11.

Discusión y conclusiones

Al analizar la matriz de transición es posible observar que las probabilidades condicionales son todas equiprobables. Esto quiere decir que estar posicionado en un determinado estado no favorece la transición a otro estado particular con mayor probabilidad, siendo los pasajes de estados independientes. En base a esto se puede concluir que la fuente es de memoria nula.

2.1.3 - Inciso c

En caso de que la fuente no fuera de memoria nula, se debe corroborar si la misma es ergódica y calcular tanto su vector estacionario como su entropía.

En el caso bajo estudio, utilizando una codificación de 4 símbolos, se obtuvo como resultado que se trata de una fuente de memoria nula. Sin embargo, se procedió a realizar el cálculo del vector estacionario a fin de poder respaldar con mayor seguridad dicha conclusión.

En primer lugar, observando que en la matriz de transición no existen ceros, es posible afirmar que se trata de una fuente ergódica, ya que la probabilidad de transición desde un estado j a cualquier otro estado i es mayor a cero, es decir, todas las transiciones entre estados están permitidas.

Una vez comprobada la ergodicidad de la fuente y dado que se trata de un conjunto finito de estados, se procede a calcular el vector estacionario.

Pseudocódigo para el cálculo del vector estacionario:

Se inicializa el vector estacionario con un valor semilla (posible ya que es ergódica)y se multiplica sucesivamente hasta que el valor se estabilice cumpliendo cierta cota de error.

```
vectorEstacionario := vector[4] = {1,0,0,0}
vectorAuxiliar := vector[4]
tolerancia = error = 10E-7
mientras que error >= tolerancia:
        vectorAuxiliar = MatrizTransicion x vectorEstacionario
        error = normaM(vectorAuxiliar, vectorEstacionario)
        vectorEstacionario = vectorAuxiliar

función normaM(vectorAuxiliar, vectorEstacionario):
        return max(abs(vectorAuxiliar - vectorEstacionario))
```

Resultados obtenidos

Vector estacionario (v*):

0.256078188441284	0.254492648233878	0.242745274004343	0.246683889320495

Tabla III - Vector de probabilidades estacionarias

Desviaciones de equiprobabilidad (Pi = 0.25):

0.0060781884412836	0.0044926482338781	0.0072547259956571	0.0033161106795044

Entropía:

H(S) = 1.99907845246353 bits

Conclusiones

Al observar la matriz de transición (y corroborar mediante el algoritmo de Warshall que todos los estados sean alcanzables) y siendo la fuente de memoria nula un caso particular de cadenas de markov (con *m=0*, dependiente de cero estados anteriores) se puede concluir que la misma es ergódica.

Dado que las probabilidades estacionarias son similares a las condicionales, se puede concluir que los estados son independientes, y por lo tanto la fuente es de memoria nula.

En una fuente de memoria nula se debe cumplir que la probabilidad condicional P(A/B) debe ser igual a P(B), dado que los eventos A y B son independientes.

Demostración:

Dado dos eventos A y B, se tiene que la probabilidad de que ocurra el evento B, habiendo ocurrido el evento A es igual a:

$$P(B/A) = \frac{P(A \cap B)}{P(A)}$$

Si los eventos son independientes

$$P(A \cap B) = P(A).\ P(B)$$

por lo que:

$$P(B/A) = \frac{P(A \cap B)}{P(A)} = \frac{P(A).P(B)}{P(A)} = P(B)$$

2.2 - Segunda parte

2.2.1 - Inciso a

Los códigos analizados en el inciso *a* de la primera sección son:

- Instantáneos
  - cumplen con la propiedad de prefijo
- Unívocamente decodificables
  - a toda secuencia de palabras de código corresponde una única secuencia de símbolos → no hay ambigüedad en la decodificación
- No singulares
  - todos los símbolos de la fuente tienen códigos diferentes
- Bloque
  - a cada símbolo le corresponde una palabra de código que permanece constante en el tiempo

2.2.2 - Inciso b

Para cada uno de los escenarios se calcularon la longitud media y la inecuación de Kraft-MacMillan como se muestra en la tabla IV.

Resultados obtenidos

	Escenario 1	Escenario 2	Escenario 3
Kraft	1.0	1.0	1.0
Longitud media	5	7	9

Tabla IV - Longitud media y valor de inecuación de Kraft para a) Escenario 1 - Palabras de código de 5 dígitos b) Escenario 2 - Palabras de código de 7 dígitos c) Escenario 3 - Palabras de código de 9 dígitos

Conclusiones

Como en cada escenario se verifica la inecuación de Kraft, es posible asegurar la existencia de un código instantáneo con las longitudes li (i=5,7,9) para cada escenario. Además, dado que los códigos propuestos verifican la condición del prefijo, se puede concluir que efectivamente los códigos presentados en cada caso son instantáneos.

Por otro lado, dado que en cada escenario los códigos presentan la misma longitud, es esperable que la longitud media del código coincida con la longitud de las palabras en cada caso.

$$L = \sum_{i=1}^{q} P_i l_i = l \sum_{i=1}^{q} P_i = l * 1 = l$$

2.2.3 - Inciso c

Para cada uno de los escenarios se calcularon el rendimiento y la redundancia como se muestra en la tabla V.

Resultados obtenidos

	Escenario 1	Escenario 2	Escenario 3
Rendimiento (%)	99.92024090216233	99.70013506647425	98.83350849442503
Redundancia (%)	0.0797590978376661	0.29986493352575394	1.1664915055749674

Tabla V - Rendimiento y redundancia para a) Escenario 1 - Palabras de código de 5 dígitos b) Escenario 2 - Palabras de código de 7 dígitos c) Escenario 3 - Palabras de código de 9 dígitos

Conclusiones

Como se puede apreciar en los resultados obtenidos, el rendimiento de los códigos para cada escenario es alto. Este resultado era el esperado ya que en cada caso la longitud media del código y la entropía coinciden con la cantidad de dígitos de las palabras código de cada escenario.

$H(S) \leq L \Rightarrow \frac{H_r(S)}{L} \leq 1$, como se verifica que $H(S) \simeq L$, entonces $\eta = \frac{H_r(S)}{L} \simeq 1$

Si bien no es posible afirmar que los códigos sean compactos, ya que implicaría verificar que la longitud media del código es igual o menor que la longitud media de todos los códigos unívocos que pueden aplicarse a la misma fuente y al mismo alfabeto, es posible afirmar, sin embargo, que son candidatos a ser elegidos como códigos compactos como consecuencia de los altos valores de rendimiento que presentan.

2.2.4 - Inciso d

Por último, se utiliza el código de Huffman para obtener un código compacto y óptimo. A continuación se detalla el pseudocódigo utilizado.

Pseudocódigo para la obtención de codificación de Huffman:

Para cada palabra generar un nodo con su símbolo y probabilidad e incluirlos a todos en una cola de prioridad cuyo criterio de prioridad es por orden creciente de probabilidad. Mientras que en la cola haya más de un elemento, desencolar dos nodos, generar un nuevo nodo cuya probabilidad es igual a la suma de las probabilidades de los nodos removidos, guardar la referencia en el nuevo nodo a los dos removidos y encolar el nuevo nodo en la cola. Cuando solo quede un elemento, retirarlo y guardarlo como la raíz del árbol binario generado.

Definición de nodo :: `Nodo {String palabra, double prob, Nodo izq, Nodo der}`

```
# genero el arbol de codificación
 pq = ColaPrioridad<Nodo>
 para cada palabra:
        pq.encolar(Nodo(palabra, probabilidad[palabra], null, null)
 mientras que tamaño(pq) > 1:
        izq = pq.desencolar()
        der = pq.desencolar()
        nodoNuevo = Nodo(null, izq.prob + der.prob, izq, der)
        pq.encolar(nodoNuevo)
```

```
raíz = pq.desencolar()

# función para generar los códigos de huffman una vez obtenido el árbol

codigosHuffman := diccionario(string, string)
función generarCodigosHuffman(Nodo nodo, String codigo):
        si nodo es hoja (izq=null y der=null):
                codigosHuffman[nodo.símbolo] = código
        sí no:
                generarCodigosHuffman(nodo.izq, código + "0")
                generarCodigosHuffman(nodo.der, código + "1")
```

Resultados obtenidos

Para cada escenario planteado se codificaron los códigos a fin de obtener los códigos de Huffman correspondientes. Se pueden consultar los archivos anexos *huffman5digitos.txt*, *huffman7digitos.txt* y *huffman9digitos.txt* para ver la calificación resultante. Los archivos reconstruidos usando la codificación obtenida son *anexo1-grupo5-huffman5.txt, anexo1-grupo5-huffman7.txt y anexo1-grupo5-huffman9.txt.* Los archivos *recovery* fueron generados a partir de los archivos reconstruidos para luego compararlos con el archivo original a fin de evaluar el correcto funcionamiento del algoritmo.

Como puede verse en esos archivos, en el escenario de 5 dígitos como las probabilidades de las palabras eran muy similares entre sí, los códigos de huffman obtenidos presentan todos la misma longitud de 5 dígitos, iguales a las longitudes originales. Como consecuencia, no se modifica el rendimiento del código ni se obtiene compresión en el archivo. Sin embargo, en los casos de 7 y 9 dígitos, como las probabilidades de las palabra progresivamente tienden a ser más diferentes entre sí, en estos casos si se obtienen códigos de huffman de diferentes longitudes, siempre buscando asignar a los símbolos con mayor probabilidad palabras código de menor longitud. En estos dos casos, por lo tanto se observa una mejora (aunque leve) en el rendimiento del código y se observa compresión al reconstruir el archivo.

	Escenario 1		Escenario 2		Escenario 3	
	Original	Huffman	Original	Huffman	Original	Huffman
Rendimiento	99.92024	99.92024	99.70013	99.75397	98.83350	99.65925
Redundancia	0.079759	0.079759	0.29986	0.246029	1.166491	0.340743
Cantidad de símbolos en archivo	31500	31500	31500	31483 (-17)	31500	31239 (-261)

3 - Conclusiones

En este trabajo pudimos corroborar todos los conceptos vistos tanto en la teoría como en la práctica. Pudimos caracterizar una fuente de información, calculando su entropía y la cantidad de

información transportada por sus símbolos, como así también sus probabilidades condicionales para verificar si la misma era o no de memoria nula Por último, caracterizamos los códigos y calculamos un código compacto y óptimo utilizando el algoritmo de Huffman.