

Teoría de la Información

Trabajo Integrador N°1

- Integrantes: Gutiérrez Paredes José María
- Email: GutierrezJoseMaria01@hotmail.com
- Fecha de entrega: 13/10
- Repositorio: <https://github.com/guteeeeeeeee/teoriaInformacion.git>

Índice

I.	Resumen	3
II.	Introducción	4
III.	Desarrollo Primera Parte	5
IV.	Desarrollo Segunda Parte	8
V.	Conclusiones	11

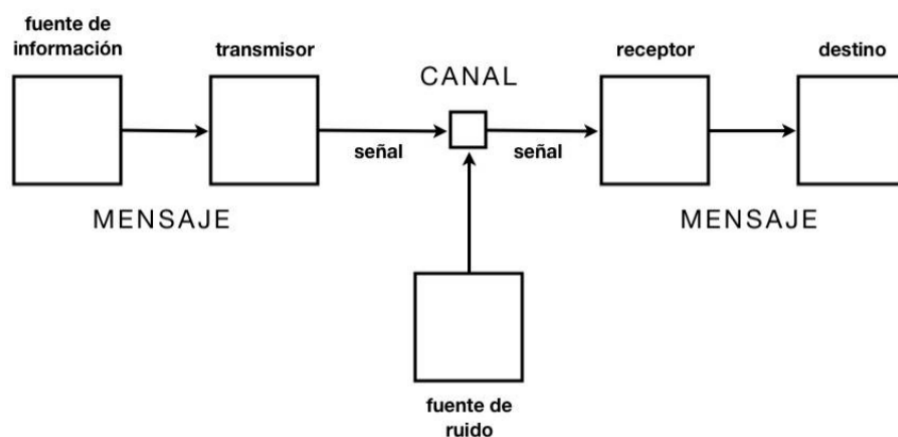
Resumen

Este trabajo consiste en asimilar todos los conceptos relacionados con la teoría de la información y la comunicación. Esta rama de las matemáticas surgió a finales de la década del 40 como una respuesta a los problemas técnicos del proceso de comunicación. En este contexto el significado del mensaje no es relevante, sino que el interés principal lo constituye todo aquello relacionado con la capacidad y fidelidad para transmitir información de los diferentes sistemas de comunicación.

Introducción

El trabajo consiste en, procesando un archivo provisto por la cátedra, plasmar todos los contenidos vistos en la materia. Para ello se desarrollarán funciones en código creadas por nuestra cuenta para calcular los valores necesarios para arribar a las conclusiones. Entre las funciones se pueden encontrar cálculo de probabilidades condicionales, probabilidades independientes, cantidad de información, entropía, entre otros, los cuales son conceptos estudiados durante la cursada. Para luego, con los resultados obtenidos, arribar a conclusiones.

Para ello primero se necesita saber qué es un modelo de comunicación (de Shannon y Weaver) representado por el siguiente diagrama.



Este modelo se podría resumir en que la fuente de información produce un cierto número de símbolos que forman el mensaje a transmitir, este a su vez pasa por un transmisor que lo codifica en señales o códigos. Se envía por un canal de comunicación el cual es el medio por el que se transportan las señales codificadas. Puede haber ruido que perturbe o modifique la señal transportada. El receptor recibe el mensaje codificado y debe ser capaz de descryptarlo para que finalmente el destinatario pueda entenderlo. Lo que se busca es un código compacto y decodificable principalmente.

Desarrollo

Una fuente de información genera en cada instante de tiempo un determinado símbolo elegido de su alfabeto (símbolos posibles) según sus probabilidades de emisión. Se puede clasificar según el rango de valores, continua si es un rango continuo de valores o discreta si es un conjunto finito de valores los que genera. Y según su relación entre los símbolos de memoria nula si los símbolos son estadísticamente independientes unos de otros y con memoria si los símbolos son dependientes de quien/quienes lo preceden.

Con el archivo provisto se obtuvieron los siguientes resultados:

```
frecuencia de cada simbolo de la fuente:
A 5263.000000 B 2927.000000 C 1810.000000
frecuencia de cada i si salio el simbolo j:
AA 2924.000000 | AB 1620.000000 | AC 719.000000 |
BA 1800.000000 | BB 62.000000 | BC 1065.000000 |
CA 539.000000 | CB 1245.000000 | CC 25.000000 |
probabilidad de cada simbolo de manera independiente:
A 0.526300 B 0.292700 C 0.181000
probabilidades condicionales:
AA 0.555577 | AB 0.553468 | AC 0.397238 |
BA 0.342010 | BB 0.021182 | BC 0.588398 |
CA 0.102413 | CB 0.425350 | CC 0.013812 |
La fuente no es de memoria nula
La fuente es ergodica
Entropia de la fuente: 0.756313 unidades trinaras/simbolo
```

Para ello se utilizó la función *procesarArchivo* la cual consiste en:

Se tiene una matriz de probabilidades condicionales y un vector de probabilidades individuales.

Abre el archivo, lee el primer carácter, aumenta su frecuencia en uno en el vector de probabilidades individuales y lo guarda como carácter anterior ya que falta otro para formar la probabilidad condicional. Además, aumenta en uno el contador de caracteres leídos.

**Lee el siguiente carácter y lo guarda como carácter actual, aumenta su frecuencia en el vector de prob. Individuales y también aumenta su frecuencia en la matriz de probabilidades condicionales en los índices i y j tales que $mat[i][j]$ es la frecuencia de que salga i (carácter actual) luego de j (carácter anterior). Se lo guarda como índice anterior, aumenta nuevamente el contador de caracteres leídos y vuelve a * iterando hasta el fin del archivo.*

Una vez calculadas todas las frecuencias operamos con la matriz de probabilidades condicionales donde a cada elemento $mat[i][j]$ se la divide por $prob[j]$ ya que la suma de las filas de cada columna debe dar uno de probabilidad. Así queda conformada la matriz de probabilidades condicionales.

Luego a cada índice del vector de probabilidades individuales $prob[i]$ se las divide por la cantidad de caracteres leídos almacenados en la variable contador para poder obtener la

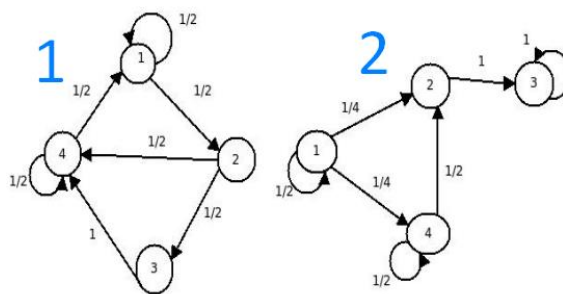
probabilidad de que salga cada símbolo de manera independiente. Así queda conformado el vector de probabilidades individuales/independientes.

Luego de obtener todas las probabilidades se puede determinar si es una fuente de memoria nula o no nula mediante la función *esMemoriaNula*.

Mientras se cumpla la condición, recorre la matriz de probabilidades condicionales verificando la igualdad de que la probabilidad de que A haya salido después de B sea la misma de que salga A de manera individual ya que no tendría que estar condicionado por el símbolo que lo precede. Si no se cumple la condición en algún caso, se termina el recorrido y devuelve False.

En este archivo se evidencia de manera clara de que la probabilidad de que B salga después de que haya salido C es casi el doble de la que tendría que tener y en el caso de que salga después de B es casi nula. Esta disparidad de probabilidades no es posible en una fuente de memoria nula por lo tanto será una fuente con memoria.

Al determinarse que se trata de una fuente con memoria se trata de determinar si se trata de una fuente ergódica o no. Una fuente ergódica consiste en que todos los símbolos son alcanzables desde otro.



En el caso de los siguientes grafos que representan a fuentes de información de Markov, el grafo 1 se trata de una fuente ergódica ya que todos los símbolos son alcanzables desde otro en cambio la fuente 2 no ya que desde el nodo 3 no puede alcanzar a ningún otro salvo a sí mismo.

Para determinarlo implementarlo en código se usa algoritmo de Marshall inicializando la matriz de pesos en 1 si hay probabilidad de que ocurra i si salió j y 99 (valor infinito) si no hay probabilidad de que ocurra.

Itera con todos los valores posibles de i y j buscando todos los posibles “puentes” k . Finalmente se muestran todas las posibles uniones entre nodos, directa o indirectamente. Es decir, si de i a j no hay probabilidad, es posible que si haya de i a k y luego de k a j .

En caso de que todos sean alcanzables, es decir, no hay caminos de peso infinito entonces devuelve true la función.

Es importante determinar si es una fuente con memoria debido a que, si se la observa durante un determinado tiempo, ésta emitirá una secuencia típica de símbolos. Al determinarse que se trata de una fuente ergódica se busca calcular su vector estacionario.

Para ello se debe restar la matriz de probabilidades con la matriz identidad y multiplicar por el vector estacionario para encontrar el vector de igual dimensión con todos valores 0.

$$\left(\begin{bmatrix} \text{MATRIZ PROB} \\ \text{CONDICIONALES} \end{bmatrix} - \begin{bmatrix} \text{MATRIZ} \\ \text{IDENTIDAD} \end{bmatrix} \right) * \begin{bmatrix} \text{Vector estacionario} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

P =	0.555577	0.553468	0.397238	
	0.342010	0.021182	0.588398	
	0.102413	0.425350	0.013812	
v =				

P²⁰⁰ (n-step transition matrix). Pⁿ vP² (n-st

P ⁿ =	0.5160244	0.5159566	0.5157474	
	0.2869584	0.2869207	0.2868043	
	0.1773729	0.1773497	0.1772777	

Siendo el vector estacionario = [0.5160244, 0.2869584, 0.1773729].

Este vector almacena las probabilidades de emisión los símbolos de la fuente cuando ésta comienza a estabilizarse/estacionarse luego de un tiempo t. No depende de las probabilidades iniciales sino de las condicionales.

Para calcular la entropía de la fuente se debe aplicar la siguiente fórmula desarrollada en la función *entropiaFuente*.

$$H_1 = \sum_i p_i * \sum_j p_{j/i} \log \frac{1}{p_{j/i}}$$

**Se recorre la matriz de probabilidades condicionales fijando la columna y acumulando por filas la prob[i][j]*(log3(1/prob[i][j])) en la variable acumParcial. Una vez acumulados todos los valores de las filas se multiplica por el vector estacionario en el índice correspondiente a la columna fijada (variable estacionaria) y se acumula en la variable acumTotal. Se incrementa la columna y vuelve a iterar *. Finalmente devuelve el acumTotal con el valor de la entropía de la fuente en unidades ternarias.*

Este valor, la entropía, puede interpretarse como el valor medio de la incertidumbre de un observador antes de conocer la salida mientras más grande sea, mayor va a ser el desconocimiento de la posible salida.

Parte 2

Resultados luego de la ejecución del programa de la segunda parte:

```
palabras codigo longitud 3

cantidad informacion: 92.331069 unidades ternarias/simbolo
entropia: 2.469192 unidades ternarias/simbolo
cumple Kraft y McMillan
longitud media: 3.000000
no es compacto
rendimiento: 82.306385% redundancia: 17.693615%

palabras codigo longitud 5

cantidad informacion: 698.651401 unidades ternarias/simbolo
entropia: 3.950588 unidades ternarias/simbolo
cumple Kraft y McMillan
longitud media: 5.000000
no es compacto
rendimiento: 79.011761% redundancia: 20.988239%

palabras codigo longitud 7

cantidad informacion: 2662.251276 unidades ternarias/simbolo
entropia: 5.246776 unidades ternarias/simbolo
cumple Kraft y McMillan
longitud media: 7.000000
no es compacto
rendimiento: 74.953939% redundancia: 25.046061%
```

Para calcular la cantidad de información se utiliza la fórmula $\sum_{i=1}^{cant\ símbolos} \log_3 \left(\frac{1}{P(S_i)} \right)$ siendo S_i cada símbolo emitido por la fuente. Es un número que nos indica la cantidad de información que aporta la fuente al receptor que la recibe (los eventos más probables serán los que menos aporten). Y para calcular la entropía se utiliza $\sum_{i=1}^{cant\ símbolos} P(S_i) I(S_i)$, es decir, la sumatoria del producto entre la probabilidad de que sea emitido cierto símbolo de la fuente y la cantidad de información que aporta ese símbolo. Puede interpretarse como el valor medio de la información por símbolo suministrada por la fuente o el valor medio de la incertidumbre del observador antes de conocer la salida.

Estos cálculos son implementados en la función procesarPalabras, una vez calculadas las probabilidades de todos los símbolos de la fuente. Se recorre nuevamente cada índice del vector para realizar los cálculos reemplazando la probabilidad por su equivalente en el vector de probabilidades y la cantidad de información aplicando su fórmula correspondiente con ayuda de la siguiente igualdad donde $\log_n(x) = \frac{\log(x)}{\log(n)}$.

La Inecuación de Kraft $\sum_{i=1}^q r^{-l_i} \leq 1$ donde q es la cantidad de palabras código, r la cantidad de símbolos diferentes que conforman el alfabeto código y l_i la longitud de cada palabra código.

Se aplica la inecuación y da como resultado que para los órdenes 3, 5 y 7 se cumple. Lo cual nos indica que es posible construir un código instantáneo con las longitudes dadas para cada símbolo. Esto no significa que cualquier código que cumpla con esas longitudes será instantáneo, se debe verificar la condición de prefijo.

Luego con la misma desigualdad, pero dándole otro significado se tiene la Inecuación de McMillan que nos dice que si se cumple la desigualdad entonces existe un código unívoco para las longitudes de código dadas.

El resultado obtenido por Kraft es lógico porque al ser todas las palabras código de la misma longitud y distintas entre sí no es posible que una sea prefija de otra. Y también serán unívocamente decodificables lo cual también es fácilmente verificable ya que para cualquier extensión finita de orden n las palabras códigos serán no singulares pues se sabe dónde empieza y termina cada palabra.

La implementación consiste en recorrer, acumulando en un variable, el vector de longitud de cada palabra código, que en cada orden tendrá siempre la misma longitud (3, 5 o 7), y elevar la cantidad de símbolos del alfabeto original (siempre será 3 ya que serán A, B y C) a la longitud negativa.

La longitud media del código dada por la siguiente expresión $\sum_{i=1}^q p_i l_i$ es el promedio del largo de cada palabra código. Está dado por la sumatoria del producto entre la probabilidad de la palabra código y su longitud. En este caso no es de mucha utilidad ya que se sabe de antemano que todas las longitudes serán fijas por lo tanto el valor de la longitud media será la cantidad caracteres que tomemos como palabras (3, 5 o 7).

Un código compacto consiste en un código instantáneo cuya 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. En términos prácticos para que un código sea compacto se debe cumplir que $P_i = \left(\frac{1}{r}\right)^{a_i}$ siendo a_i un número entero indicando la longitud que debe tener la palabra código, r la cantidad de símbolos distintos del alfabeto.

Se encuentra implementado con un ciclo que itera mientras se cumpla la condición de que el largo de cada palabra código sea igual a $\log_r \left(\frac{1}{P_i}\right)$, siendo r el número de símbolos distintos de la fuente, en el caso de que recorra todo el vector de probabilidades entonces se encontró un código compacto ya que cumple con todas las longitudes si no devuelve false.

En los resultados obtenidos en ningún código se obtuvo un código compacto esto es debido a que al haber probabilidades dispares entre sí y que todas tengan la misma longitud no se está aprovechando para hacer códigos donde las más probables sean más cortas y las menos probables más largas logrando así obtener un mayor ahorro en el almacenamiento y en la transmisión.

La eficiencia o rendimiento de un código se define como $\frac{H(S)}{L}$ siendo $H(S)$ la entropía de la fuente y L la longitud media de la fuente. Se puede interpretar como una manera de indicar la cantidad de información importante con las longitudes usadas. A mayor cantidad de información relevante en menor longitud media este cociente se acercará a 1 que sería el 100% de rendimiento. También se encuentra la redundancia que se define como $1 - \text{eficiencia}$ y nos indica que hay información innecesaria que nos puede dificultar interpretar el mensaje codificado. Se busca máximo rendimiento y mínima redundancia.

Los resultados arrojaron que el rendimiento cae a medida que aumenta el largo de las palabras código. Esto es lógico debido a que al ir aumentando la longitud de los códigos aumenta la cantidad de palabras código que casi no se emiten lo que produce que su aporte a la entropía sea nulo y esto provoca que disminuya el rendimiento (las palabras con probabilidad 0 de ser emitidas o cercanas al casi no aportan a la entropía). El rendimiento sería máximo en el caso de que la probabilidad de emisión de cada palabra código fuera equiprobable entonces la entropía sería máxima y daría igual a la longitud media dando como resultado un rendimiento del 100%.

Para codificar los símbolos se utilizó el método Shannon-Fano el cual es un método subóptimo cuyo fin es construir un código óptimo de acuerdo a las probabilidades de cada palabra código.

Para su implementación primero se tuvo que ordenar mediante el algoritmo Quicksort el vector de probabilidades (junto con el índice al que correspondía) de cada palabra código.

Entra al método recursivo con unos determinados i y j , donde comienza y donde termina esa segmentación del vector respectivamente. Luego se lo divide al vector en 2 partes buscando de que la diferencia entre la suma de las probabilidades entre ambas partes sea mínima. En el caso de que se encuentre una probabilidad que al agregarla a un subgrupo aumente esa diferencia no la agrega y corta el recorrido con los subgrupos divididos. Una vez encontrada esa separación, a uno se le asigna 1 y al otro 0. Y manda a llamar nuevamente al método desde donde comienza el vector hasta la mitad de la primera parte y también desde el siguiente de la mitad de la primera parte hasta donde termina el vector.

Una vez terminado de ejecutar se puede observar en los archivos que lo que busca es otorgarles la menor longitud a las palabras código más frecuentes buscando así reducir el costo en la transmisión y el almacenamiento de los códigos.

Conclusiones

Como conclusión se puede decir es un trabajo integrador interesante ya que se plasmaron todos los temas vistos hasta el momento en la materia combinando la teoría con la práctica. Durante la realización de este trabajo se comprendió que si bien un código puede cumplir con la inecuación de Kraft no es condición suficiente para afirmar que sea instantáneo. También que los valores eficiencia y redundancia son importantes ya que se puede ver que tanto aporta información valiosa un código en relación con la longitud de sus palabras. Se aprendió a calcular las longitudes que debe tener cada palabra código de acuerdo a su probabilidad y cantidad de símbolos fuente para que éste sea compacto.

Se arribó a la conclusión de que un buen código compacto es importante ya que así habrá un mayor ahorro en el almacenamiento y en la transmisión para poder reducir el ruido que puede ocurrir en el canal de transmisión y que éste no afecte a la transmisión del mensaje.

A la hora de codificar con Shannon-Fano se evidenció que para los símbolos con mayor probabilidad de emisión se buscan códigos con longitudes más pequeñas o reducidas mientras que para los que poseen casi nula probabilidad su código es más grande en comparación.