Teoría de la Información

**Trabajo Integrador N° 1:**

**Unidad I a Unidad IV hasta Rendimiento y Redundancia de un código**

**Grupo 6**

**Integrantes:**

**Aguilera Marcos** [**marcos.aguilera.7.13@gmail.com**](mailto:marcos.aguilera.7.13@gmail.com)

**Castorina Matías** [**matiascastorina@gmail.com**](mailto:matiascastorina@gmail.com)

**Noseda Demian** [**nosedademian@gmail.com**](mailto:nosedademian@gmail.com)

**// INDICE**

**Resumen**

En este informe vamos a tratar con fuentes de información de memoria nula y fuentes de Markov, mediante el uso de algoritmos desarrollados por nosotros e implementados en Java para realizar las operaciones correspondientes.

**Introducción**

Durante el desarrollo del trabajo se tratará con fuentes de memoria nula, para las mismas se requiere calcular la cantidad de información de cada símbolo y la entropía de la fuente. También se calculará un código instantáneo para ellas y se generarán secuencias simuladas de símbolos. Todo esto utilizando como dato la distribución de probabilidades de la fuente.

Trataremos también con fuentes de memoria de Markov, para dichas fuentes calcularemos su entropía, vector de estado estacionario y podremos generar una secuencia simulada de símbolos a partir de la distribución de probabilidades dada por su matriz de transición.

**Fuentes de memoria nula**

Se trata de una fuente de información donde la aparición de cada símbolo no depende del símbolo anterior obtenido, formalmente se dice que los símbolos son estadísticamente independientes.

**Cantidad de información**

Es la forma de cuantificar la información aportada en la observación de un suceso probabilístico, esta se calcula en base a su probabilidad y utilizamos un algoritmo basado en la lógica del pseudocódigo descripto a continuación.

*Resultado = 0*

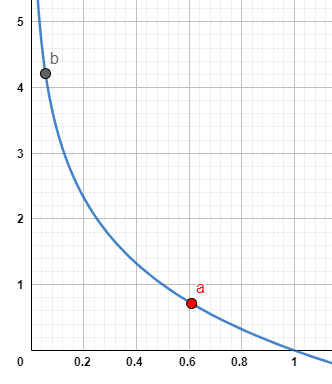
*Si la probabilidad es mayor que cero entonces*

*Resultado = (-1 \*log (probabilidad) / log (base))*

*Devolver resultado*

Utilizaremos para ejemplificar una fuente de memoria nula binaria de cuatro símbolos. Tendremos en cuenta la probabilidad de un símbolo “a” es 0.61 al aplicar este algoritmo el programa da como resultado que la cantidad de información es 0.71312 bits. Luego si calculamos la cantidad de información del símbolo “d” con probabilidad 0.054 obtenemos que la cantidad de información es 4.2109 bits.

Podemos notar que el símbolo menos probable aporta mas cantidad de información, si lo analizamos mediante el grafico de la cantidad de información en función de la probabilidad vemos el siguiente resultado:

Se puede ver que la curva es asintótica a cero por lo que a medida que la probabilidad se acerque a este número, la cantidad de información aumentará. De la misma forma a medida que la probabilidad se acerca a 1 la cantidad de información se aproxima a cero siendo nula cuando la probabilidad alcanza su máximo valor en 1.

**Entropía (fuente de memoria nula)**

La entropía es el nombre que se le da a la cantidad media de información por símbolo de la fuente.

Utilizamos un algoritmo para calcularla que está representado en el siguiente pseudocódigo:

*n= cantidad de símbolos*

*Vector= vector de probabilidades y cantidad de información por símbolos*

*Mientras posición actual sea menor a n hacer*

*Resultado= resultado + probabilidad en la posición actual del vector \* cantidad de información en la posición actual del vector*

*Posición actual = posición actual + 1*

*Fin del mientras*

*Devuelve resultado*

Para el cálculo de la entropía de las Fuentes de memoria nula A y C (ver en apéndice) se obtuvieron los siguientes resultados:

Entropía de la fuente A=1.51648

Entropía de la fuente C=2.56308

Teniendo en cuenta que la cantidad de símbolos de la fuente A es 4 y el de la C es 7, podemos apreciar que a mayor cantidad de símbolos es esperable una mayor entropía. Para contrastar este resultado si utilizamos la fuente B de 5 símbolos se obtiene una entropía de 2.08179.

Esto es lógico en general ya que la suma de las probabilidades asociadas a los símbolos deben sumar uno, entonces cuantos más símbolos posea una fuente, sus probabilidades asociadas tenderán a ser menores para poder cumplir con dicha suma, así aumentando la cantidad de información aportada por cada uno de ellos, y finalmente aumentando la también la entropía de la fuente. No obstante esto no puede ser considerado una regla ya que en ciertos casos esto no se cumplirá, por ejemplo comparando una fuente de dos símbolos equiprobables contra una de cuatro símbolos donde la probabilidad de uno de ellos es mucho mayor que la del resto, anexamos en el apéndice una tabla de resultados comparativos a modo de contraejemplo.

**Generador de código instantáneo**

A la hora de generar un código instantáneo para una fuente de memoria nula de base 2 consideramos una solución basada en la siguiente secuencia de código:

0, 10 , 110 , 1…0

Elegimos esta secuencia dado que el primer símbolo es 0 y como el resto de los símbolos comienzan en 1 ningún símbolo será prefijo de otro (se usa 0 como separador). Se asigna al símbolo de mayor probabilidad el código más corto, continuando de manera descendente en probabilidad y ascendente en longitud de código.

Pseudocódigo de la función:

*Vector= vector de símbolos*

*Ordena vector por probabilidad descendente*

*Código en la primera posición del vector=”0”*

*i=2*

*Mientras que i sea menor que la cantidad de símbolos hacer:*

*Código en la posición i del vector= i veces “1” y “0” (ejemplo i=2 “110”)*

*i = i +1*

*Fin del mientras*

*FIN*

# Cálculo de longitud media del código

Realizamos este cálculo para una fuente de memoria nula que posea una codificación. En nuestro caso la codificación es generada por nuestro mismo programa. Se trata de una media de longitudes de cada código ponderadas por la probabilidad.

Pseudocódigo de la función:

*Vector= vector de símbolos (contiene probabilidad y códigos)*

*Sumatoria = 0*

*I=0*

*Mientras que i sea menor que la cantidad de símbolos hacer*

*Sumatoria= sumatoria+probabilidad del vector en i +longitud del código del vector en i*

*Fin del mientras*

*Devolver sumatoria*

Ejemplo de uso:

Usamos una fuente de memoria nula “D” (ver en apéndice) cuya codificación es: S1=0, S2=1110, S3=10, S4=110

El resultado obtenido de la longitud media es 2. Teniendo en cuenta que la entropía de la fuente es 1.84644 bits podemos ver que cumple la condición que la entropía es menor igual a la longitud de cualquier código instantáneo para la fuente (Sabemos que se trata de un código instantáneo dado que lo da el mismo programa).

# Cálculo de la inecuación de Kraft

El cumplimiento de esta inecuación es una condición necesaria para que una codificación con cierta longitud de código para cada símbolo sea instantánea.

Utilizaremos dicha ecuación para comprobar si los códigos generados por el programa la cumplen (todos deberían dado que estamos generando codificaciones instantáneas). El siguiente Pseudocódigo representa la función que realiza el cálculo

*Vector= vector de símbolos (contiene probabilidad y códigos)*

*Sumatoria = 0*

*Base= base de la fuente de memoria (en general usamos 2)*

*I=0*

*Mientras que i sea menor que la cantidad de símbolos hacer*

*Sumatoria= sumatoria+base elevado a la longitud del código en la posición i del vector*

*Fin del mientras*

*Devolver sumatoria*

Este pseudocódigo devuelve el primer término de la inecuación de Kraft el cual debe cumplir con ser menor o igual a 1 para verificar dicha inecuación

Ejemplo de uso:

Volviendo a la fuente de memoria nula “D” (ver en apéndice) tenemos una longitud de código de 1 , 2 , 3 , 4 , para cada símbolo respectivamente. El resultado obtenido de la inecuación de Kraft es la siguiente: 0.9375<=1.

Podemos apreciar que cumple con la inecuación de Kraft de la forma esperada.

# Verificación de código compacto

Un código compacto es aquel en el que las longitudes de las palabras cumplen con ser iguales al siguiente entero mayor a la cantidad de información de dicho símbolo

*Techo= función que devuelve el siguiente entero mayor de un numero real*

*Vector= vector de símbolos (contiene probabilidad y códigos)*

*Condición =verdadero*

*I=0*

*Mientras que i sea menor que la cantidad de símbolos y condición sea verdadero hacer*

*Condición= Techo de (cantidad de información del vector en i) es igual a longitud del código del vector en i*

*Fin del mientras*

*Devolver condición*

Este pseudocódigo devuelve si para un vector que representa una fuente y posee una codificación dicha codificación es compacta.

Utilizando a la fuente de memoria nula “D” (ver en apéndice) con longitudes de código 1 , 2 , 3 , 4 , para cada símbolo respectivamente. El resultado obtenido es que el código no es compacto. Es lógico que la condición no se cumpla dado que la función utilizada para generar el código instantáneo no fue hecha teniendo en cuenta elegir longitudes de palabra que cumplan con la condición de compacto.

**Conclusión**