Guion I
Razón de Compresión y Entropía

Información sobre la entrega de la práctica

Las prácticas se entregarán en un único fichero comprimido Practica01ApellidoNombre.zip. El fichero contendrá:

Las funciones de Matlab a realizar en ficheros .m con los nombres de las funciones que se indiquen en el guion.
Los trozos de código a realizar, que se entregarán todos en los pasos correspondientes de un único fichero .m llamado Practica01ApellidoNombre.m . Este fichero lo crearás modificando el fichero .m Practica01MolinaRafael.m en el servidor.
Las discusiones y respuestas solicitadas en el guion se entregarán en un único fichero pdf. El nombre del fichero será Practica01ApellidoNombre.pdf. Lo construirás editando Practica01MolinaRafael.doc y salvándolo en formato pdf.

El objetivo de esta práctica es afianzar los conceptos básicos de codificación y compresión de datos: factor de compresión y entropía y estudiar la importancia de la modelización de una fuente para obtener buenos resultados en la compresión de los datos generados por esa fuente. Necesitaremos ficheros de datos para comprimir. Hemos preparado un conjunto de datos de diferentes tipos de datos y características que se encuentran en la carpeta *Datos* dentro de *MATERIAL PRÁCTICAS*.

Trabajaremos con el concepto de entropía de una fuente como medida de la información de la misma y veremos como un correcto modelado de los datos es importante para obtener los mejores resultados de un sistema de compresión.

Aplicaremos los conceptos de entropía y modelado de los datos a imágenes y ficheros de texto y binarios.

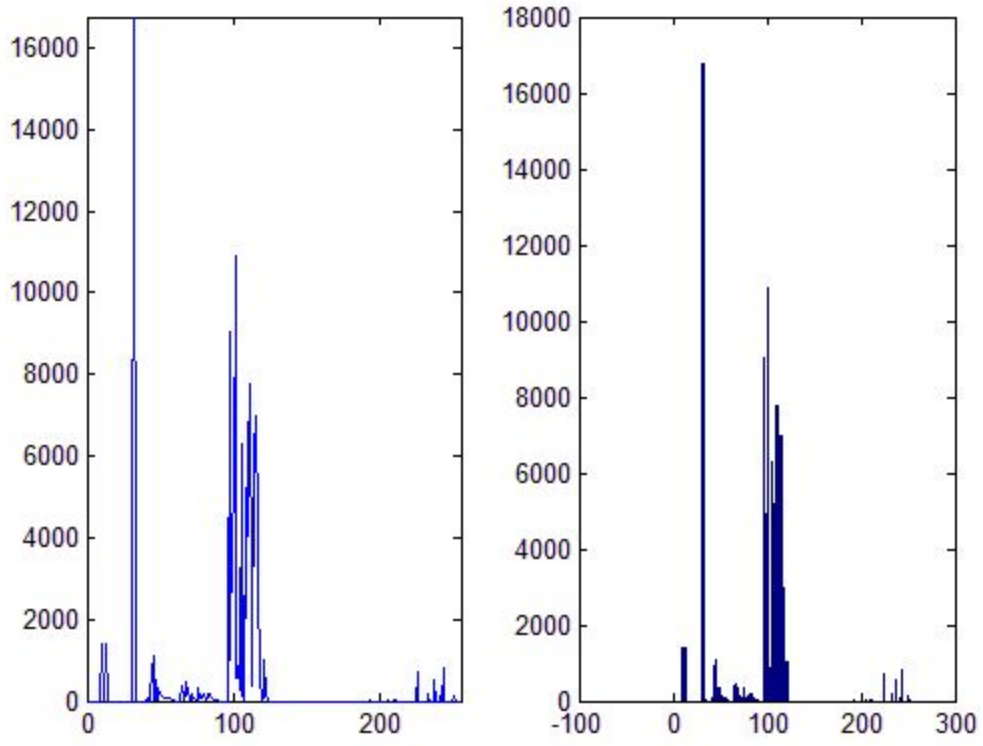
Es importante distinguir entre el número de letras que tiene nuestro alfabeto y la representación de las mismas en el fichero. Suponemos aquí que cada letra se almacena inicialmente usando una codificación que asigna el mismo número de bits a cada una de las letras del alfabeto. Aunque tengamos el alfabeto {0, 1}, nosotros usaremos generalmente un byte para representar cada letra (codificada en binario, usando su código ASCII o cualquier otro sistema de representación) aunque está claro que un sistema más compacto sería usar sólo 1 bit por letra. Esta representación a un byte por letra ha sido escogida simplemente por comodidad puesto que realizar los programas usando bytes es más sencillo y cómodo que tener que usar representaciones a nivel de bits de los mismos.

Paso 1

Limpiamos el espacio de trabajo. Leemos los datos del fichero 'constitución española.txt', calculamos el número de veces que aparece cada carácter y dibujamos el correspondiente histograma.

Entiende que hacen las funciones que utilizamos y sus parámetros. No incluyas ningún camino para el fichero, garantízate que has definido bien el path de Matlab. Observa el tipo de dato que leemos del fichero, éste será por defecto el tipo que usaremos.

```
clear all; close all;
fichero='constitucion española.txt';
fid=fopen(fichero, 'r')
[words count]=fread(fid,inf,'*uint8');
fclose(fid)
histograma= histc(words,[0:255]);
subplot(1,2,1);
plot([0:255],histograma); axis('tight')
% si prefieres puedes usar la función bar
subplot(1,2,2); bar([0:255],histograma)
```



Paso 2

Como ves el histograma está lejano de ser uniforme en el intervalo [0,255] con lo que es seguro que necesitaremos menos de 8 bits por dato. Vamos ahora a calcular la entropía de la fuente.

Crea una función que se llame **entropiaTUSINICIALES**, en mi caso sería **entropiaRMS** de mi nombre **R**afael **M**olina **S**oriano, que acepte un histograma, calcule su distribución de probabilidad asociada y devuelva la entropía. Nota: En Matlab lo podemos hacer de una forma muy sencilla. Primero convertimos el histograma en una distribución de probabilidad que podemos llamar prob. A continuación calculamos la entropía pero tenemos que hacerlo utilizando sólo aquellos términos con prob >0 (lee el manual de la función **find**). Matemáticamente esto no es necesario pero a Matlab no le gusta hacer 0*log2(0). Luego escribe la fórmula de la entropía.

Es importante que escribas una buena implementación de entropía. Debes evitar los for y debe servir para cualquier tamaño de alfabeto, no solo aquellos que tienen 256 símbolos como máximo.

Incluye aquí el código de tu función entropiaTUSINICIALES.m

```
function [resultado]= entropiaJMRC(histograma)

     suma= sum(histograma);

     prob= histograma./suma;

     prob(find(prob==0.0))=[];

     resultado=-sum(prob.*log2(prob));

end
```

Paso 3

Ejecuta la función que calcula la entropía, en mi caso entropiaRMS,

```
H= entropiaRMS(histograma)
```

Debe salirte 4.4880.

Paso 4

1. ¿Qué significa el valor de la entropía que has obtenido?.
2. ¿Cuál sería el factor de compresión que obtendríamos si usamos un modelo de codificación que alcanzase la entropía?.

3. ¿Podremos, a lo largo del curso, ganar a la entropía?

Escribe tus respuestas aquí

1. *El valor obtenido significa que cada dato en vez de ser representado por 8 bits, se puede representar por 4.48*
2. *8*tam_imagen/4.4880*tam_imagen=1,78*
3. *Se podrá ganar a la entropia obtenida anteriormente usando otras tecnicas como teniendo en cuenta que los valores tienen relación entre ellos, aunque el concepto como entropía nunca podrá ser mejorado*

Paso 5

Vamos a limpiar de nuevo el espacio de trabajo y las imágenes que hemos mostrado. A continuación leemos la imagen camera.pgm

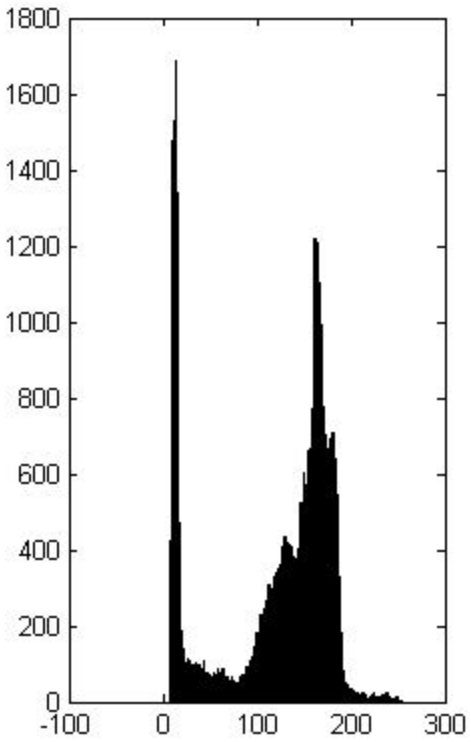
```
clear all; close all;
A=imread('camera.pgm');
% Mostramos la imagen camera.pgm
subplot(1,2,1); imshow(A);
```

Paso 6

A continuación de la imagen A, no del fichero camera.pgm, calculamos el histograma, lo mostramos y calculamos la entropía de la fuente. Observa cómo introducimos la imagen bidimensional en la función histc

```
histograma=histc(A(:),[0:255]);
subplot(1,2,2); bar([0:255],histograma)
entropiaRMS(histograma)
```

Obtenemos gráficamente



Paso 7

1. ¿Cuál es el valor de la entropía que has obtenido?.
2. ¿Cuál sería el factor de compresión que obtendríamos si usamos un modelo de codificación que alcanzase la entropía?.

Escribe tus respuestas aquí

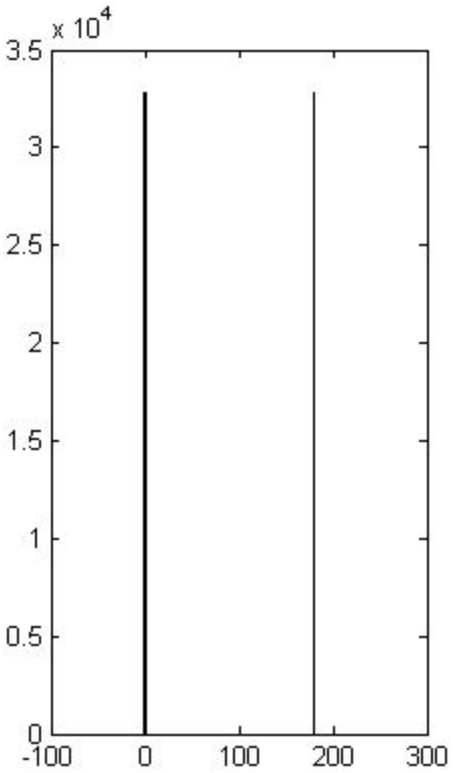
1. 7,0097
2. 8*tam_imagen/7,0097*tam_imagen=1,14

Paso 8

Limpia el espacio de trabajo y las variables. Creamos una imagen de tamaño 256x256 con niveles de gris 0 (fondo) y un cuadrado con nivel de gris 180. Utiliza el tipo uint8. Calcula ahora el histograma de esta imagen y su entropía.

```
clear all; close all;
A=uint8(zeros(256));
A(1:128,:)=uint8(180);
imshow(A)
histograma= histc(A(:),[0:255]);
figure; bar([0:255],histograma)
```

```
H=entropiaRMS(histograma)
```



Paso 9

1. ¿Qué significa el valor de la entropía que has obtenido?.
2. ¿Cuál sería el factor de compresión que obtendríamos si usamos un modelo de codificación que alcanzase la entropía?.
3. ¿Podremos, a lo largo del curso, ganar a la entropía?

Escribe tus respuestas aquí

1. El resultado obtenido es 1, lo que indica que un dato como minimo se puede representar con 1 bit.
2. 8*tam_imagen/1*tam_imagen=8
3. Si debido a que no estamos teniendo en cuenta la relación que mantienen los datos entre ellos

Paso 10

1. Si hicieras más grande (y luego más chico) el cuadrado blanco, ¿qué le pasaría a la entropía?
2. ¿Cuánto valdría la entropía si toda la imagen fuera blanca o negra?

3. ¿Qué significaría el valor de la entropía obtenido en este caso?.

Escribe tus respuestas aquí

1. La entropia se reduce en ambos casos.
2. La entropría valdría 0.
3. Que no se necesita almacenar nigun dato en la matriz porque todos son iguales.

Paso 11

La entropía de segundo orden de una fuente con un alfabeto de *m* letras se obtiene considerando la secuencia como formada por parejas de letras, (X_1, X_2). Supuesto que las parejas de letras (X_1, X_2) son independientes e idénticamente distribuidas, la entropía de segundo orden se calcula de acuerdo a la siguiente fórmula:

$$H(S) = - \sum_{=1}^{m} \sum_{j=1}^{m} P(X_1 = i, X_2 = j) \log P(X_1 = i, X_2 = j).$$

Por tanto, debemos calcular las probabilidades $P(X_1 = i, X_2 = j)$. Para guardar estas probabilidades podemos construir una matriz de tamaño *m x m* en que cada celda *(i,j)* contiene el número de veces que la pareja $(X_1 = i, X_2 = j)$ aparece en la secuencia dividido entre el número de parejas de letras en la secuencia.

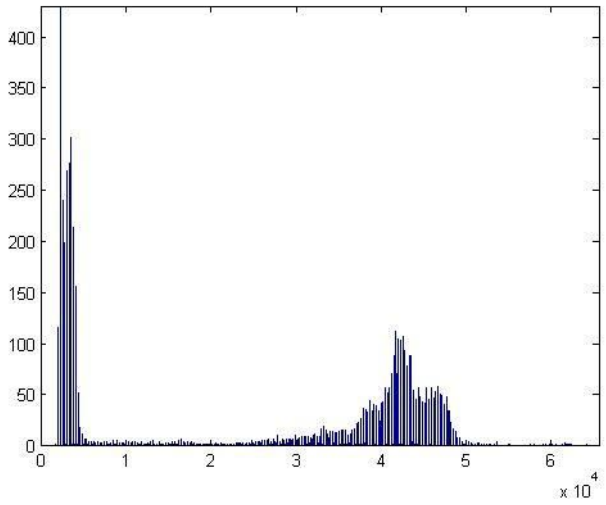
Esta matriz de probabilidades también la podemos representar en forma de un vector, *V*, con *m x m* elementos en que cada posición del vector $k = i*m+j$ contiene la probabilidad $P(X_1 = i, X_2 = j)$ y calcular la entropía como

$$H(S) = - \sum_{k=1}^{mxm} V(k) \log V(k).$$

Vamos a leer los caracteres de dos en dos y luego calcularemos la entropía. Lo haremos con el fichero 'camera.pgm'. Observa la sintaxis de lectura y como calculamos el histograma

```
clear all; close all;
fichero='camera.pgm'
fid=fopen(fichero, 'r')
words=fread(fid,inf,'*uint16');
fclose(fid)
histograma= histc(words,[0:256*256-1]);
bar([0:256*256-1],histograma), axis('tight')
H=entropiaRMS(histograma)
```

El histograma es



Paso 12

¿Cuál es la entropía de esta fuente que codifica los símbolos de 'camera.pgm' de dos en dos?.

1. ¿Qué significa el valor de la entropía que has obtenido?.

Escribe tus respuestas aquí

1. La entropía vale 11.12
2. Que cada dato se podrá representar con 11.12 bits

Paso 13

Leamos ahora el mismo fichero pero de byte en byte y calculamos la entropía

```
clear all; close all;
fichero='camera.pgm'
fid=fopen(fichero, 'r')
words=fread(fid,inf,'*uint8');
fclose(fid)
histograma= histc(words,[0:255]);
H=entropiaRMS(histograma)
```

Paso 14

1. Compara los valores de la entropía que has obtenido en los pasos 11 y 13. ¿Qué está pasando?

Escribe tus respuestas aquí

1. La entropía ahora ha pasado a valer 7.01, los valores que en el ejercicio 11 estaban de dos, ahora son considerados individuales. Por tanto si en el ejercicio 11 se representaba 2 datos con aproximadamente 11.12 ahora dos datos serían representados con aproximadamente 14.02

Paso 15

Una vez que conocemos los programas que vamos a usar para calcular la entropía, vamos a aplicarlos a diferentes tipos de datos y ver su significado.

Para los ficheros bird.pgm, ptt1.pbm, texto10000.txt, Cinco semanas en globo - Julio Verne.txt completa la siguiente tabla. ¿Qué significan los valores que obtienes?. Escribe el código correspondiente en el Paso 15 del fichero Practica01ApellidoNombre.

Escribe tus respuestas aquí

Fichero	Entropía de primer orden	Entropía de segundo orden
Bird.pgm	6.77	10.25
ptt1.pbm	0.69	1.12
texto10000.txt	0.99999	1.99998
Cinco semanas en globo - Julio Verne.txt	4.4747	7.87

Significado:

Tanto Bird, ptt1 como Cinco semanas en globo se puede ver que la entropia de segundo orden es mejor que la de primera. En cambio el texto10000 es justo el doble, esto se debe a que estan representados 0 y 1 sin ninguna dependencia entre los datos, por tanto para representar uno de los datos es necesario 1bit y para dos datos es necesario 2 bits.

Paso 16

Lee la siguiente imagen

```
A=imread('bird.pgm');
```

Paso 17

En el paso 17 del fichero Practica01ApellidoNombre.m escribe código para

1. Calcular la entropía de la matriz que contiene la imagen.
2. Calcular la diferencia de cada píxel con el anterior por filas. Es decir, vamos a calcular A(i,j)-A(i,j-1). No debes usar bucles y además tienes que tener mucho cuidado con las diferencias ya que la diferencia de dos caracteres sin signo da un carácter sin signo y esto no es lo que queremos hacer. Para calcular la diferencia de la primera columna considera que la columna anterior es cero.
3. Dibujar en una misma ventana la imagen de diferencias y su histograma.

4. Calcular la entropía de primer orden sobre la imagen de diferencias.

Nota 1: Ten cuidado con los tipos de datos cuando hagas diferencias

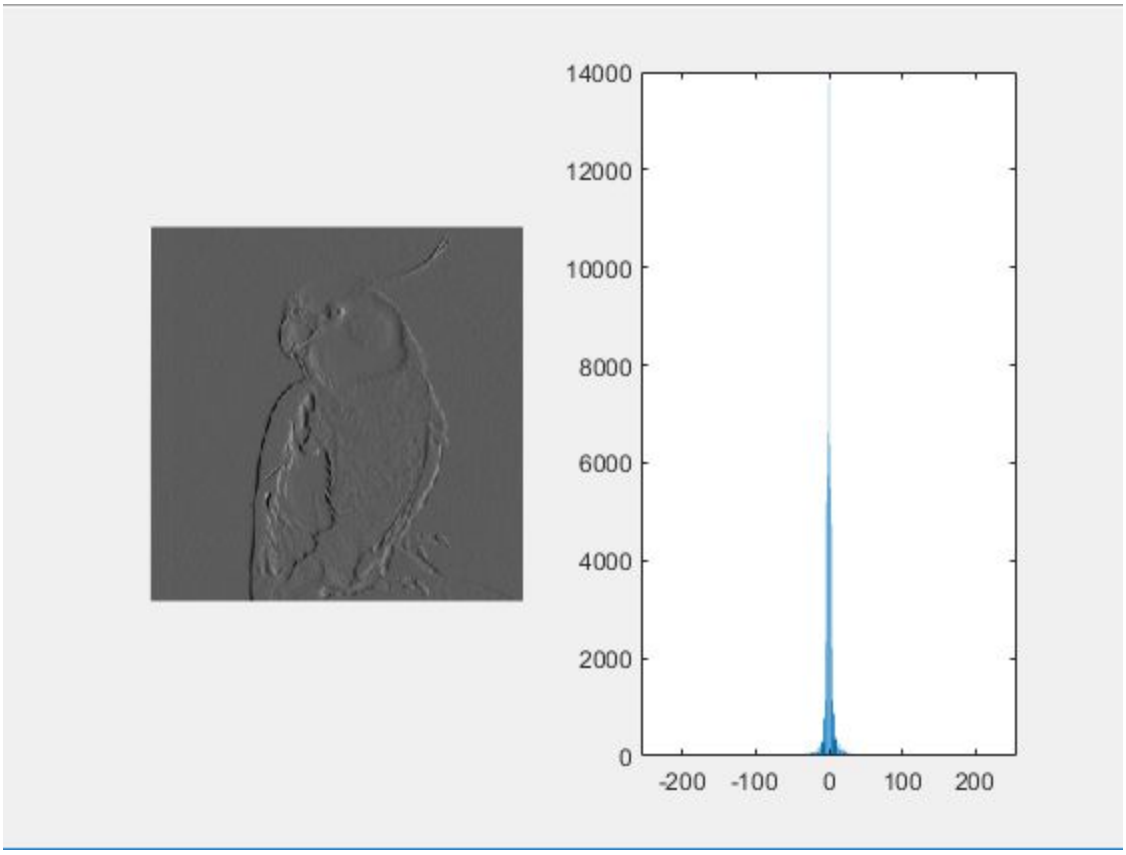
Nota 2: ¡Cuidado al hacer las diferencias! Para una imagen en escala de grises, las diferencias pueden estar en el intervalo [-255, 255] y por tanto necesitamos al menos 9 bits para representarlas.

Paso 18

1. Incluye el gráfico del paso 17 aquí
2. ¿Cuál son las entropías de la imagen original y de la imagen de diferencias?.
3. Compáralas y explica el resultado

Escribe tus respuestas aquí

1.



2. De la imagen original la entropia es 6.7 mientras que la de diferencias es 4,2333

3. El resultado es que tenemos una matriz que puede tener valores negativos, donde la mayoria de los datos tienen valor 0 o muy cercano al 0(como podemos observar en el diagrama) ya que existe una gran dependencia de datos.

Paso 19

Supongamos que tenemos una fuente que obtiene palabras de cuatro letras. Supongamos además que las letras son generadas aleatoriamente suponiendo una distribución uniforme sobre las 27 letras del abecedario. ¿Cuántos bits necesitaríamos en media para representar cada palabra de cuatro letras?

Escribe tus respuestas aquí

$\sum_{1}^{27}$(1/27 * log(27))= 4.7588*4letras= 19.0195500087

Paso 20

Forma ahora M=100 palabras de N=4 letras del alfabeto. Utiliza el siguiente código de Matlab

```
clear all
letras=['a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i' 'j' 'k' 'l' 'm' 'n' 'o' 'p' ...
    'q' 'r' 's' 't' 'u' 'v' 'w' 'x' 'y' 'z' 'á' 'é' 'í' 'ñ' 'ó' 'ú' 'ü']

elem_alf=numel(letras);
M=100;  %Numero de palabras
N=4;  % letras en cada palabra

rng(0); % Mira para qué sirve esta orden
codigosMxN=randsrc(M,N,[1:elem_alf;ones(1,elem_alf)/elem_alf]);
palabrasMxN=letras(codigosMxN)

fid = fopen('palabras4letrasgeneradasuniformemente.txt','w');
for i=1:M
    fprintf(fid,'%s\n',palabrasMxN(i,:) );
end
fclose(fid);
```

Paso 21

1. Utiliza un procesador de textos o aspell -l para comprobar las palabras incorrectas. Inclúyelas aquí.

Escribe tus respuestas aquí

áfvb ñámw ekáb órrc uflr dtód jiíá svsá úwux úyte fogv ücjr úhpü qóhv áféá eégo nrho óüfé áchc úooe vdkf búóm íaoé ózgá wáñb zíün ycor mndn viiv fánu xotj bóio jgta bixü dehf éedd wíjm ktkg úsnq beql oícú muió zláb áqay gnói qcyn ohqs vetú xghn yhpü jnúj wbsx vñrv fúhr dqqx qquv úlwf lñne tmmü hdüf yzbb imñs qhóñ xnáw ñddg úeim súlp eúwü esef ibxí éhdv ilvm áéqg iazo óbxp lfñd gvñt iylh uvxm pogt lsbi éjyj tyqu sgpi ówñé jguü ymuy yuíl mzát sctd cógñ bzhñ rqñá zoai óoqt ekfa sqüo pqxk aáqf lápf

Paso 22

Una fuente que tiene probabilidad 1/2 de generar una vocal (5 vocales) y también 1/2 de generar una consonante (22 consonantes) y una vez que sabes que es una vocal todas tienen la misma probabilidad y lo mismo ocurre con las consonantes ¿Cuál sería la entropía de esta fuente?.

Escribe tu respuesta aquí

Siendo P(v) las probabilidades de las vocales cuyo valor es 1/2 * 1/5 que resulta 1/10 mientras que la P(C) es de 1/2 * 1/22 que es 1/44, esto sin saber la letra anterior, lo que significa la primera letra de todas las palabras, mientras que la siguientes letras tienen una probabilidad igual a todas de 1/27.

Aplicando la fórmula $\sum_{1}^{5}(1/10 * \log(10)) + \sum_{1}^{22}(1/44 * \log(44)) = 4{,}38996405$

Paso 23

El siguiente trozo de código lee palabras de cuatro letras del diccionario de la RAE y las almacena en una hilera que contiene todos los caracteres y los convierte a caracteres sin signo

```
clear all;
fid = fopen('PalabrasDRAE.4letras.txt', 'r');
letras=fscanf(fid,'%s');
letras2num= uint16(letras); %observa la conversión de los dos bytes
```

Paso 24

Escribimos ahora el código en Matlab que calcula la estimación de la probabilidad de cada letra del abecedario castellano usando las frecuencias con la que aparecen en el fichero PalabrasDRAE.4letras.txt.

```
histograma=histc(letras2num,[0:256*256-1]); %observa el rango
indicespos=find(histograma>0);
probabilidades=histograma(indicespos)/sum(histograma(indicespos));
```

Paso 25

Escribe en el paso 25 de Practica01tuapellidonombre.m código en Matlab que forme ahora 100 palabras de cuatro letras usando las probabilidades calculadas. Sálvalas en el fichero palabras4letrasgeneradasRAE.txt. Inicializa el generador de números aleatorios usando rng(0).

Paso 26

4. ¿Qué palabras has obtenido?
5. ¿Cuántas de las palabras que has obtenido en el paso anterior tienen sentido?,
6. ¿Cuáles son?,
7. Compara los resultados obtenidos con los del paso 21.

Escribe tu respuesta aquí

1. saoa urgo aesa umma oafm aoya edts nons zoop ároa ajco íaem zclí luco sats asbj imci uúas raca ziia oaea aávh tais yrbs osua rtíi raim hiai oddo asio oioe audi eboa adpñ aaca saaa oteg eneb znil aale itaz godv rfsa rmap bivd lapi icln oany pbci rcjñ eiáe oano oumo azcm allo lloo zeob euha ogfú caña rraa dhun lcuu oiro uaab zadg nyej azoñ anaa dapt tcao dfog sslb cari vapl faua bouo dpec ooog ljbo fnad sere orlo nbld uous eboñ rgop rote grso nana avbu arcu mlus riad yilo aeaa nlíi llpe asla erlb

2. 7 palabras
3. umma, luco, cari, orlo, rote, nana
4. los cambios que existen entre el resultado del ejercicio 21 y este son evidentes, en el ejercicio 21 ninguna de las palabras están reconocidas por la RAE mientras que este hay 7. En las palabras de este ejercicio podemos también apreciar que están mejor organizadas debido a que se tiene en cuenta las probabilidades, mientras que el 21 son totalmente aleatorias.