Aplicación para calcular la entropía de bits en iOS y Android

García García Jonathan Eduardo
jgarciag1404@alumno.ipn.mx

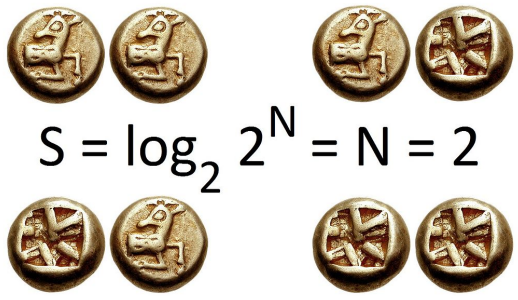
Abstracto—Este documento es el reporte técnico, explicativo del proceso por el cúal se consiguio crear una aplicación que ejemplifique forma simple y didactica el calculo de la entropía de bits o entropía de Shannon

I. Fundamento teórico

En teória de la información, la entropía de una variable aleatoria es el promedio de la ”información”, ”sorpresa” o ”incierta” inherente a los posibles resultados de la variable. El concepto the entropía de la información fue introducido por en su artículo de 1948 ”A Mathematical Theory of Communication” y frecuentemente es llamada entropía de Shannon en su honor.
Como ejemplo, considere una moneda con probabilidad p de caer en cara y probabilidad 1-p de caer en cruz.La entropía máxima es para p = 1/2, cuando no hay razón para esperar un resultado sobre otro, y en este caso, el lanzamiento de una moneda tiene una entropía de un bit. La entropía mínima es cuando p = 0 o p = 1, cuando el evento es conocido y la entropía es cero bits. Otros valores de p dan diferentes entropías entre cero y uno bits.
Fórmula:

$$H(X) = -\sum P(x_i) \log P(x_i)_{i=1}^{n}$$

Ejemplo.
Dada una variable discreta aleatoria , con posibles resultados , que ocurren con probabilidad la entropía de esta formalmente definida como: donde Sigma denota la suma de los posibles valores de la variable y log es el logaritmo, la elección de la base varía entre las diferentes aplicaciones. La base 2 da la unidad de bits (o ”shannons”), mientras que la base e da las ”unidades naturales” nat, y la base 10 da una unidad llamada ”dits”, ”bans” o ”hartleys”.



II. Desarrollo

A. Plataformas:

1) iOS
2) Android

B. Framework:

Se opto por utilizar una tecnología de Desarrollo multiplataforma pues la interfaz es muy sencilla ademas de el poco tiempo de desarrollo y el deseo de abarcar dos plataformas donde y obtener exactamente el mismo resultado.
A pesar de esto se programarón funciones especificas siguiendo las reglas de cada plataforma como son:

- Acceso a la camara
- Acceso a la galeria de fotos
- Acceso a el sistema de archivos
- Vinculación de librería nativa escrita en C++ para el cálculo rápido y eficiente de la entropía

C. Calculando la entropía

Se opto por crear una librería dinámica en el caso de Android y una librería estática para el caso de ios ambas escritas en C++ Este procedimiento es llamado desde el

```
void CalculateTotalEntropy(EntropyLibrary* handle)
{
    handle->TotalEntropy = 0;
    unsigned long TotalSymbols = 0;
    for (unsigned long i = 0; i < handle->Symbols.size(); i++)
    {
        Symbol* symbol = handle->Symbols[i];
        TotalSymbols += symbol->Count;
    }

    for (unsigned long i = 0; i < handle->Symbols.size(); i++)
    {
        Symbol* symbol = handle->Symbols[i];
        long double count = symbol->Count;
        symbol->Frecuency = count / TotalSymbols;

        handle->TotalEntropy += (symbol->Frecuency * log2(1 / symbol->Frecuency));
    }
}
```

código de alto nivel para obtener el resultado de la entrópia y los simbolos, así como las probabilidades de cada uno mediante el uso de punteros.
Al final de el cálculo toda la memoria ”insegura” es propiamente liberada. Igualmente se tiene acceso a una

```
public static float Calculate(params float[] Probabilities)
{
    IntPtr array = Marshal.AllocHGlobal(sizeof(float) * Probabilities.Length);
    Marshal.Copy(Probabilities, 0, array, Probabilities.Length);
    float ent = (float)Math.Round((EntropyLibraryWrapper.Calculate(array, Probabilities.Length) * 100));
    if (ent > 100)
    {
        ent = 100;
    }
    return ent;
}
```

libreria gráfica para el calculo de histograma y entropía por imagen

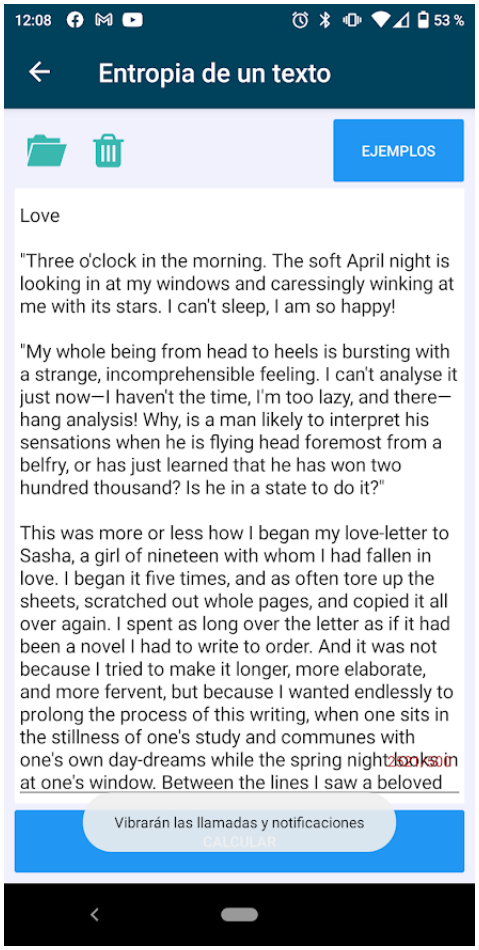
```
1 reference | Jonathan Eduardo Garcia Garcia, 12 days ago | 1 author, 1 change
public static async Task<PictureHystogram> CalculateEntropy(CachedImage FileInfo)
{
    await Task.Yield();
    PictureHystogram hystogram = new PictureHystogram();
    try
    {
        var png = new Hjg.Pngcs.Chunks.PngChunkIHDR(new Hjg.Pngcs.ImageInfo(cols:10, rows:10, bitdepth:8, alpha:true));
        using (var memory = new MemoryStream(await FileInfo.GetImageAsJpgAsync()))
        {
            memory.Position = 0;
            BitmapFactory factory = new BitmapFactory();
            factory.AddCodec(new BitmapCodec());
            factory.AddCodec(new PngCodec());
            factory.AddCodec(new JPEGCodec());
            factory.AddCodec(new TGACodec());
            using (Image<Pixel> result = factory.Decode(memory))
            {
                using (Image<Pixel> bitmap = result.GetBitmap(x:0, y:0, w:result.Width, h:result.Height))
                {
                    await hystogram.Red.Calculate(result, GetByte:(x:int, y:int) => bitmap.Get(x, y).R, Color:Pixels.Red);
                    await hystogram.Green.Calculate(result, GetByte:(x:int, y:int) => bitmap.Get(x, y).G, Color:Pixels.Green);
                    await hystogram.Blue.Calculate(result, GetByte:(x:int, y:int) => bitmap.Get(x, y).B, Color:Pixels.Blue);
                    hystogram.TotalEntropy = await EntropyOfImage(bitmap);
                }
            }
        }
    }
    catch (Exception ex)
    {
        Log.Logger.Error(ex, messageTemplate:"BarcodeDecoding.GetImage");
        await Tools.Instance.CustomMessageBox.Show(ex.Message);
    }
    return hystogram;
}
```

```
3 references | Jonathan Eduardo Garcia Garcia, 4 days ago | 1 author, 2 changes
internal async Task Calculate(Image<Pixel> result, Func<int, int, byte> GetByte, Pixel Color)
{
    await Task.Yield();
    var Hystogram = new int[256];
    int MaxValue = 0;
    for (int i = 0; i < result.Width; i++)
    {
        for (int j = 0; j < result.Height; j++)
        {
            int value = GetByte.Invoke(i, j);
            Hystogram[value]++;
            if (MaxValue < Hystogram[value])
                MaxValue = Hystogram[value];
        }
    }

    int histHeight = 128;
    Bitmap img = new Bitmap(w:histHeight, h:128);
    using (Graphics g = Graphics.FromImage(img))
    {
        g.Clear(p0:Pixels.Transparent);
        float fmax = (float)MaxValue;
        for (int i = 0; i < Hystogram.Length; i++)
        {
            var pct:float = Hystogram[i] / fmax;   // What percentage of the max is this value?
            if (pct >= 0.8999)
            {
                continue;
            }

            g.DrawLine(Color,
                p0:new Point(x:i, y:img.Height),
                p0:new Point(x:i, y:img.Height - (int)(pct * histHeight))  // Use that percentage of the height
            );
        }
    }
```

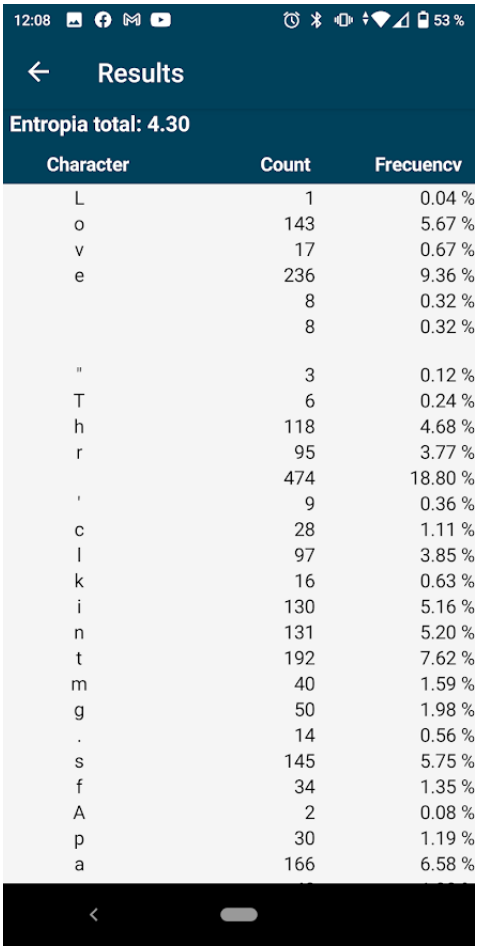
D. Capturas de pantalla de código

1) Pantalla principal

2) Pantalla para ingresar texto y calcular la entropía

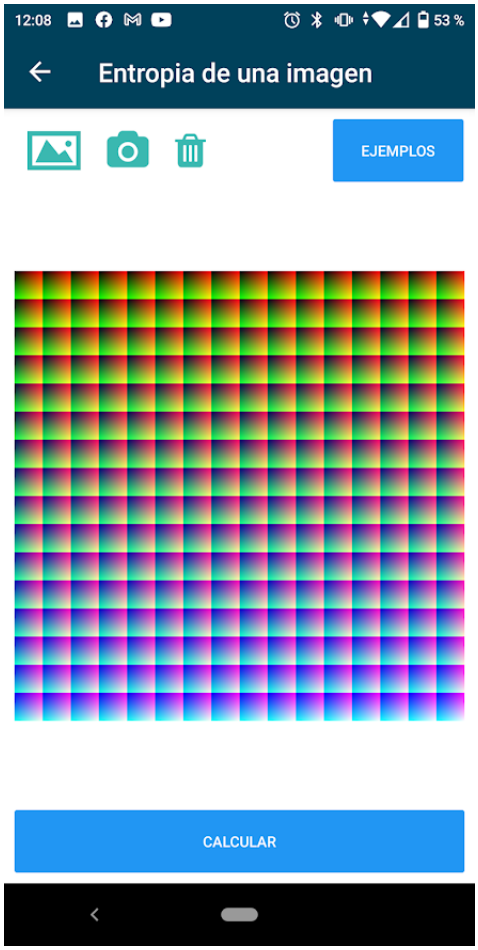


3) Resultados del de la entropía del texto

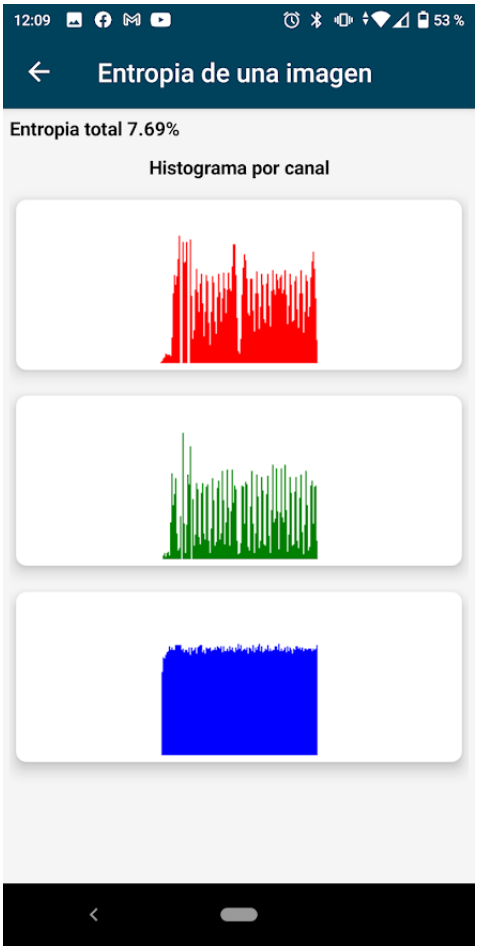


Character	Count	Frecuencv
L	1	0.04 %
o	143	5.67 %
v	17	0.67 %
e	236	9.36 %
	8	0.32 %
	8	0.32 %
"	3	0.12 %
T	6	0.24 %
h	118	4.68 %
r	95	3.77 %
	474	18.80 %
'	9	0.36 %
c	28	1.11 %
l	97	3.85 %
k	16	0.63 %
i	130	5.16 %
n	131	5.20 %
t	192	7.62 %
m	40	1.59 %
g	50	1.98 %
.	14	0.56 %
s	145	5.75 %
f	34	1.35 %
A	2	0.08 %
p	30	1.19 %
a	166	6.58 %

4) Entropía de una imagen



5) Resultados del de la entropía de una imagen



6) Fundamento teórico

7) Relación entre la probabilidad de dos eventos y la entropía total

III. Disponible como ”Shannon Entropy” en:

Download on the App Store

AppStore

GET IT ON Google Play

PlayStore

GitHub

El código puede consultarse en Github