

Como Programar una red Neuronal de Reconocimiento de Captchas

20 de Agosto del 2006

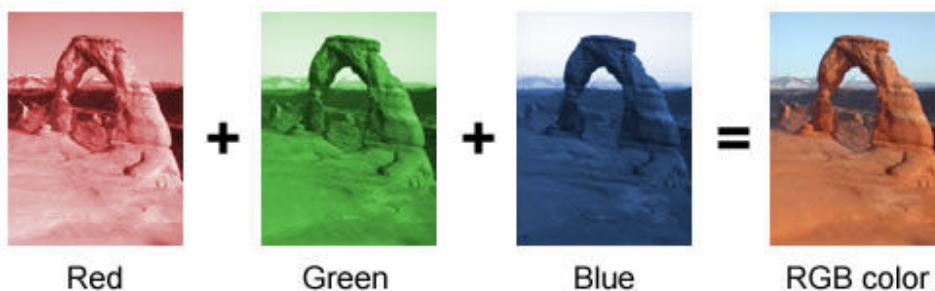
Introducción

Bueno regreso con este nuevo manual para explicar como funcionan las cosas en verdad para poder leer un captcha, primeramente hay que dejar en claro que cosa tenemos enfrente, es una imagen no un archivo de texto, es absurdo soñar que puedes abrir la imagen como texto y buscar algo que sirva, lo mas importante aquí es darnos cuenta con que estamos trabajando.

Antes que nada quiero agradecerle a Megabyte en buen plan porque el me dio la idea al intentar hackear mi sitio, este research lo hice gracias a el. Mientras le mostraba a alguien el código me hicieron un comentario muy gracioso *"Tu eres como el melate: haces que las ilusiones sucedan"*, así es la idea fue de el, pero aquí expongo la técnica de cómo es que es posible reconocer y leer las imágenes.

Comprendiendo la imagen

Básicamente aquí lo que tenemos que tomar en mente es que estamos tratando con imágenes que constan de una paleta tridimensional que va de 256 a 16.5 millones de colores, y al mismo tiempo cada imagen consta de 4 canales de los cuales los mas conocidos son solo 3 RGB (rojo, verde y azul) estos 3 canales constan de valores de entre 0 y 255, el cuarto canal es el Alpha, este maneja solo dos tonos blanco y negro, y maneja lo opaco de cada color encima de los 3 canales pasados, sus valores varían de 0 a 127.



Para empezar hay que tomar en cuenta que no estamos seguros del cuarto canal en las imágenes, no sabemos si lo tengan o no (no es necesario), aunque si lo trajeran nos ahorran mucho en el código que vamos a escribir. Así que usaremos los 3 canales conocidos y buscaremos un captcha sobre el cual trabajar, cada lector de captchas tiene que estar diseñado en especifico para cada tipo de captcha, poco a poco iremos viendo como se hace esto, e igual podemos programar una red neuronal que aprenda de varios captchas a la vez, pero en este manual solo hablaremos de como hackear los captchas de Security Nation (el código sirve para cualquier captcha y cualquier variación con unos ajustes).



Trabajaremos sobre esta imagen, a simple vista hasta para nosotros es difícil de leer contiene el texto **TPKHC**, pero que pasos hay que seguir para que un software lo reconozca.

Paso Uno – Limpiar el fondo

Lo primero que hay que hacer para limpiar la imagen es leerla en memoria:

```
$img = imagecreatefromjpeg($imageurl);  
Limpia($img);
```

Como verán una vez cargado en memoria paso la imagen por la función limpia que esta listada y detallada a continuación:

```
function limpia ($image){  
  
    //aqui tomo el alto y el ancho de la imagen  
  
    $w = imagesx($image);  
    $h = imagesy($image);  
  
    //estos dos ciclos for anidados son para barrer pixel por pixel de la imagen x,y  
  
    for ($x = 0; $x <= $w; $x++){  
        for ($i = 0; $i <= $h; $i++){  
  
            //la función de abajo obtiene los valores numéricos del color de cada pixel y los da a $r,$g,$b  
            //NOTA: para que un captcha sea legible al ojo humano el texto tiene que contar con colores separados  
            //numéricamente en el grafico tridimensional de la paleta por un mínimo de 20 a 100, de lo contrario se mezclaría con  
            //el ruido de la imagen, y seria imposible de leer.  
  
            $rgb = imagecolorat($image, $x, $i);  
            $r = ($rgb >> 16) & 255;  
            $g = ($rgb >> 8) & 255;  
            $b = $rgb & 255;  
  
            //Una vez con los valores debemos de tomar en cuenta una cosa mientras más altos más blancos,  
            //mientras mas bajos mas oscuros, si analizamos la imagen en un editor grafico veremos que los  
            //colores que forman las letras que son negras varían de 0,0,0 a 50,50,50, así que dividimos estos  
            //colores y pintamos cada píxel debajo de estos valores de negro 0,0,0 y los superiores de blanco  
            //255,255,255, técnicamente deberían quedar las letras sueltas en un grafico B/N  
  
            if (($r > 55) || ($g > 55) || ($b > 55)){  
                imagesetpixel($image, $x, $i, imagecolorallocate($image, 255, 255, 255));  
            }else{  
                imagesetpixel($image, $x, $i, imagecolorallocate($image, 0, 0, 0));  
            }  
        }  
    }  
  
    return $image;  
}
```

Veamos como queda el captcha que estamos trabajando después de esta función:



Bien como podemos apreciar hemos completado casi con éxito el primer paso, aun queda ruido en la imagen por un detalle, existen pixeles que con la mezcla de canales de colores, no pueden pintarse de blanco sin embargo buscan un color alterno normalmente el mas cercano en la representación dimensional de la paleta. Pero a final de cuentas esta mucho mejor que antes.

Segundo Paso - Separar Caracteres

Una vez semi limpia nuestra imagen tenemos que separar cada carácter en una imagen aparte, un OCR reconoce caracteres no palabras de caracteres, así que con un editor grafico tomamos las coordenadas, y creamos la imagen separada de cada letra con el código PHP a continuación:

```
//como verán repito la operación 5 veces ya que es un captcha de 5 letras
//aquí creamos en una variable una imagen $width y $height tienen un valor de 15
```

```
$imgchar1 = imagecreate($width,$height);
$imgchar2 = imagecreate($width,$height);
$imgchar3 = imagecreate($width,$height);
$imgchar4 = imagecreate($width,$height);
$imgchar5 = imagecreate($width,$height);
```

```
//luego copiamos cada letra de la imagen original a cada una de nuestras imágenes nuevas.
```

```
imagecopy($imgchar1, $img, 1, 1, 4, 10, 10, 12);
imagecopy($imgchar2, $img, 1, 1, 13, 10, 10, 12);
imagecopy($imgchar3, $img, 1, 1, 22, 10, 10, 12);
imagecopy($imgchar4, $img, 1, 1, 31, 10, 10, 12);
imagecopy($imgchar5, $img, 1, 1, 40, 10, 10, 12);
```

```
//Para trabajar mas fácil con estas imágenes las guardamos temporalmente en el disco duro
//Podemos usarlas directamente, pero así es mas fácil de explicar. De lo contrario lidiaríamos con
//una respuesta de pixeles blancos o negros no deseados en memoria.
```

```
imagejpeg($imgchar1,"chr1.jpg");
imagejpeg($imgchar2,"chr2.jpg");
imagejpeg($imgchar3,"chr3.jpg");
imagejpeg($imgchar4,"chr4.jpg");
imagejpeg($imgchar5,"chr5.jpg");
```

```
//aquí creamos la imagen nuevamente con esta función creada también
```

```
$ch1 = crearimagenfile("chr1.jpg");
$ch2 = crearimagenfile("chr2.jpg");
$ch3 = crearimagenfile("chr3.jpg");
$ch4 = crearimagenfile("chr4.jpg");
$ch5 = crearimagenfile("chr5.jpg");
```

```
//la función simplemente detecta que tipo de imagen es y regresa su alto, su ancho y la imagen en si.
```

```
function crearimagenfile($img_file){
    global $width,$height;
    $img=0;
    $img_sz = getimagesize( $img_file );
    switch( $img_sz[2] ){
```

```

case 1:
$img = ImageCreateFromGif($img_file);
break;
case 2:
$img = ImageCreateFromJpeg($img_file);
break;
case 3:
$img = ImageCreateFromPng($img_file);
break;
}
$width = $img_sz[0];
$height = $img_sz[1];
return $img;
}

```

Bien ahora ya tenemos caracteres sueltos ahora que? Ahora hay que regresarnos un poco y crear la base de conocimiento de la aplicación.

Paso Tres – Enseñarle a Leer

Un bebe no sabe leer si no hasta que le enseñan, así que es lo mismo que haremos con la aplicación, primeramente tomando muestras de cada letra y guardándola ya limpia con el mismo código del paso uno, (deberá tomar solo unos minutos).

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

Ahora bien ya que tenemos los caracteres por separado tenemos que pasarlos a un formato comparable, es lógico que no se pueden comparar las imágenes, pero si podemos comparar los colores de cada canal de los pixeles de la imagen así que creamos lo que yo llamo un Carácter Datamap que se ve como se muestra a continuación:

```

T@:0,0,0,1,0,0,2,0,0,3,0,0,4,0,0,5,0,0,6,0,0,7,0,0,8,0,0,9,0,0,10,0,0,11,0,0,12,0,0,13,0,0,14,0,0,15,1:1,0,0,1,1,0,1,2,0,1,3,0,1,4,0,1,5,0,1,6,0,1,7,0,1,8,0,1,9,0,1,10,0,1,11,0,1,12,0,1,13,0,1,14,0,1,15,1:2,0,0,2,1,0,2,2,1,2,3,0,2,4,0,2,5,0,2,6,0,2,7,0,2,8,0,2,9,0,2,10,0,2,11,0,2,12,0,2,13,0,2,14,0,2,15,1:3,0,0,3,1,0,3,2,1,3,3,0,3,4,0,3,5,0,3,6,0,3,7,0,3,8,0,3,9,0,3,10,0,3,11,0,3,12,0,3,13,0,3,14,0,3,15,1:4,0,0,4,1,1,4,2,1,4,3,0,4,4,1,4,5,0,4,6,0,4,7,0,4,8,0,4,9,0,4,10,0,4,11,0,4,12,0,4,13,0,4,14,0,4,15,1:5,0,0,5,1,0,5,2,1,5,3,1,5,4,1,5,5,1,5,6,1,5,7,1,5,8,1,5,9,1,5,10,1,5,11,1,5,12,0,5,13,0,5,14,0,5,15,1:6,0,0,6,1,0,6,2,1,6,3,1,6,4,1,6,5,1,6,6,1,6,7,1,6,8,1,6,9,1,6,10,1,6,11,1,6,12,0,6,13,0,6,14,0,6,15,1:7,0,0,7,1,0,7,2,1,7,3,0,7,4,0,7,5,0,7,6,0,7,7,0,7,8,0,7,9,0,7,10,0,7,11,0,7,12,0,7,13,0,7,14,0,7,15,1:8,0,0,8,1,0,8,2,1,8,3,0,8,4,0,8,5,0,8,6,0,8,7,0,8,8,0,8,9,0,8,10,0,8,11,0,8,12,0,8,13,0,8,14,0,8,15,1:9,0,0,9,1,0,9,2,1,9,3,0,9,4,0,9,5,0,9,6,0,9,7,0,9,8,0,9,9,0,9,10,0,9,11,0,9,12,0,9,13,0,9,14,0,9,15,1:10,0,0,10,1,0,10,2,0,10,3,0,10,4,0,10,5,0,10,6,0,10,7,0,10,8,0,10,9,0,10,10,0,10,11,0,10,12,0,10,13,0,10,14,0,10,15,1:11,0,0,11,1,0,11,2,0,11,3,0,11,4,0,11,5,0,11,6,0,11,7,0,11,8,0,11,9,0,11,10,0,11,11,0,11,12,0,11,13,0,11,14,0,11,15,1:12,0,0,12,1,0,12,2,0,12,3,0,12,4,0,12,5,0,12,6,0,12,7,0,12,8,0,12,9,0,12,10,0,12,11,0,12,12,0,12,13,0,12,14,0,12,15,1:13,0,0,13,1,0,13,2,0,13,3,0,13,4,0,13,5,0,13,6,0,13,7,0,13,8,0,13,9,0,13,10,0,13,11,0,13,12,0,13,13,0,13,14,0,13,15,1:14,0,0,14,1,0,14,2,0,14,3,0,14,4,0,14,5,0,14,6,0,14,7,0,14,8,0,14,9,0,14,10,0,14,11,0,14,12,0,14,13,0,14,14,0,14,15,1:15,0,0,15,1,0,15,2,0,15,3,0,15,4,0,15,5,0,15,6,0,15,7,0,15,8,0,15,9,0,15,10,0,15,11,0,15,12,0,15,13,0,15,14,0,15,15,0

```

Como verán este datamap muestra la información de la imagen de la letra “T”, tiene el nombre al inicio porque sabíamos que letra era y lo vamos a enseñar, el código después de la @ si se fijan esta formado por secciones de 3 números 0,0,0 el primero determina el píxel en la posición X, el segundo el píxel en la posición Y, y el tercero especificado en blanco y negro para nosotros representa 0 color blanco, 1 color negro. Este datamap se crea con el siguiente código:

```

function crearchardatamap($image,$name){

//obtenemos los valores de las variables ya declaradas alto y ancho

global $width,$height;

//especificamos que lo primero en la variable es el nombre de la letra

```



Ahora como ya tenemos nuestros 5 datamaps vamos a la parte del reconocimiento, por una parte es similar a como el hombre reconoce caracteres y no es nada complicada. Cuando a nosotros nos enseñaron las letras las vimos analizamos y creamos dentritas de datamaps en el cerebro, al software le hemos hecho lo mismo, cuando nosotros vemos una tipografía no muy común, primero el cerebro compara cada letra con las letras que hemos aprendido que existen y la asociamos con la mas parecida. Así que en esta ocasión haremos lo mismo.

El reconocimiento consta en comparar cada uno de los 256 pixeles con los que consta cada letra, es decir el datamap del carácter del captcha contra los datamaps de las 27 letras que le hemos enseñado, una ves hecho esto, la letra que corresponde a la imagen es la que menos pixeles de diferencia tenga, y en dado caso que tuviera diferencias mínimas tenemos que decirle que aprenda la variación de la letra al programa. Así sabrá comparar mas fácil y rápido el captcha. Para esto creamos el siguiente código:

```
// primero mandamos al datamap creado del captcha de cada letra a una función junto con
// el archivo temporal
```

```
$texto .= charletter($datamap1,"chr1.jpg");
$texto .= charletter($datamap2,"chr2.jpg");
$texto .= charletter($datamap3,"chr3.jpg");
$texto .= charletter($datamap4,"chr4.jpg");
$texto .= charletter($datamap5,"chr5.jpg");
```

```
//ahora tenemos que recordar los caracteres que ya conocemos.
//sacamos la lista de archivos de datamaps
```

```
$archivos = filelist();
```

```
//si no existen conocimientos previos, iniciamos en el sistema la función esta mas abajo y le
// Enseñamos las letras básicas.
```

```
if (empty($archivos)){
    echo "Hola Gracias por dejarme nacer!<br>";
    iniciarsistema(); //referenciada mas abajo
    echo "<br>Red De Inteligencia activa, 27 caracteres en mi memoria.";
    Exit;
}
```

```
//de lo contrario hacemos un arreglo con los nombres de cada archive de datamap
```

```
$archivos = substr($archivos, 0, strlen($archivos)-1);
$archivos = explode(",",$archivos);
```

```
//la función que compara las letras
```

```
function charletter($datamap1,$file){
    global $archivos;
```

```
//declaramos que no hay errores
```

```
    $errores =0;
```

```
//hacemos un ciclo donde abrimos cada datamap de conocimiento
```

```
    for ($i=0;$i<count($archivos);$i++){
        $data = base64_decode(str_rot13(file_read($archivos[$i])));
        $data = explode("@",$data);
        $data = substr($data[1], 1, strlen($data[1]));
```

```
$data = explode(":",$data);
$numdata = count($data);
```

//aquí hacemos un ciclo for dentro del ciclo donde comparamos el datamap de la letra con
//el datamap del archive actual pixel por pixel, en caso de que el pixel no coincide el
//numero de errores aumenta por 1.

```
for ($x=0;$x<=$numdata-1;$x++){
    if (($data[$x]) != ($datamap1[$x])){$errores++;}
}
```

//guardamos en un arreglo la cantidad de errores y en otro la letra con la que la comparo, limpiamos
//la variable errores

```
$erroresa[$i] = $errores;
$erroresb[$i] = $archivos[$i];
$errores = 0;
}
```

//aquí seleccionamos el valor del arreglo que menos errores tenga, es decir el valor mínimo y
//obtenemos su numero de index, numero que corresponde a la letra en el arreglo de letras.

```
$value = min($erroresa);
$key = array_search($value, $erroresa);
$letra = base64_decode(str_rot13(file_read($erroresb[$key])));
$letra = explode("@",$letra);
```

//si el valor mínimo de errores es diferente de 0 significa que es una letra similar pero no exacta a
//como la conocía así que le decimos que la aprenda con la función learnchar que esta abajo

```
if ($value != 0){learnchar($file,$letra[0]);}
return $letra[0];
}
```

//ESTA FUNCION LE ENSEÑA AL PROGRAMA A RECONOCER EL ABECEDARIO BASICO

//estaba mas arriba y aquí la pongo, llama a la función learnchar, con las letras del muestreo que sacamos de la a - z

```
function iniciarsistema(){
    learnchar("muestreo/A.JPG","A");
    learnchar("muestreo/B.JPG","B");
    learnchar("muestreo/C.JPG","C");
    learnchar("muestreo/D.JPG","D");
    learnchar("muestreo/E.JPG","E");
    learnchar("muestreo/F.JPG","F");
    learnchar("muestreo/G.JPG","G");
    learnchar("muestreo/H.JPG","H");
    learnchar("muestreo/I.JPG","I");
    learnchar("muestreo/J.JPG","J");
    learnchar("muestreo/K.JPG","K");
    learnchar("muestreo/L.JPG","L");
    learnchar("muestreo/M.JPG","M");
    learnchar("muestreo/N.JPG","N");
    learnchar("muestreo/O.JPG","O");
    learnchar("muestreo/P.JPG","P");
    learnchar("muestreo/Q.JPG","Q");
    learnchar("muestreo/R.JPG","R");
    learnchar("muestreo/S.JPG","S");
    learnchar("muestreo/T.JPG","T");
    learnchar("muestreo/U.JPG","U");
    learnchar("muestreo/V.JPG","V");
    learnchar("muestreo/W.JPG","W");
}
```

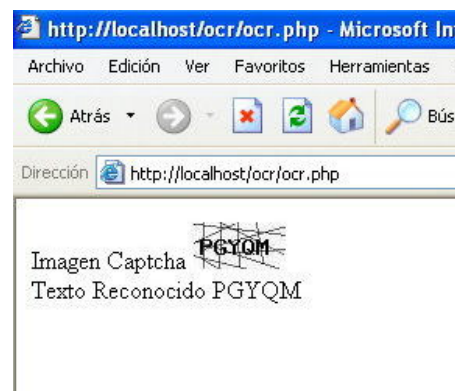
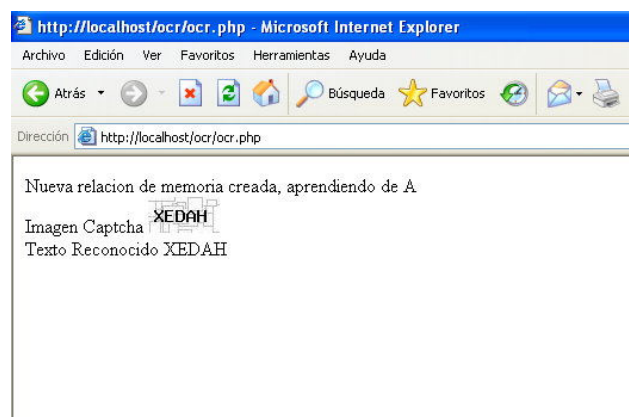


```
learnchar("muestreo/X.JPG","X");
learnchar("muestreo/Y.JPG","Y");
learnchar("muestreo/Z.JPG","Z");
unlink("muestreo/A.JPG");
unlink("muestreo/B.JPG");
unlink("muestreo/C.JPG");
unlink("muestreo/D.JPG");
unlink("muestreo/E.JPG");
unlink("muestreo/F.JPG");
unlink("muestreo/G.JPG");
unlink("muestreo/H.JPG");
unlink("muestreo/I.JPG");
unlink("muestreo/J.JPG");
unlink("muestreo/K.JPG");
unlink("muestreo/L.JPG");
unlink("muestreo/M.JPG");
unlink("muestreo/N.JPG");
unlink("muestreo/O.JPG");
unlink("muestreo/P.JPG");
unlink("muestreo/Q.JPG");
unlink("muestreo/R.JPG");
unlink("muestreo/S.JPG");
unlink("muestreo/T.JPG");
unlink("muestreo/U.JPG");
unlink("muestreo/V.JPG");
unlink("muestreo/W.JPG");
unlink("muestreo/X.JPG");
unlink("muestreo/Y.JPG");
unlink("muestreo/Z.JPG");
rmdir("muestreo");
}
```

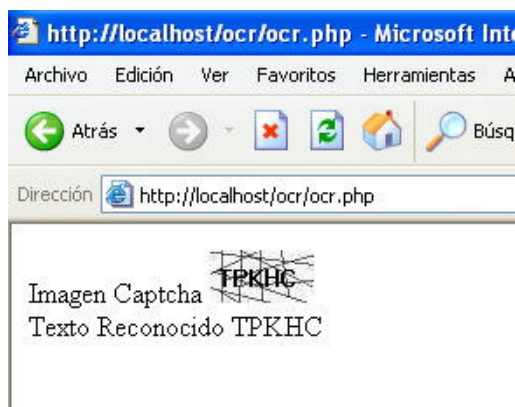
//con esta función guarda en su memoria, como es la letra nueva que le enseñamos o que aprendió
//simplemente obtiene el datamap de la imagen y lo guarda.

```
function learnchar($image,$name){
    global $width,$height;
    $imagen = crearimagenfile($image);
    $datamap = str_rot13(base64_encode(crearchardatamap($imagen,$name)));
    guardar($datamap,$name);
    echo "Nueva relacion de memoria creada, aprendiendo de $name <br>";
}
```

Una vez hecho esto en las primeras ocasiones tendremos un programa como el que se muestra a continuación:



Y con el tiempo mientras más variaciones aprenda reconocerá más rápido, abajo la imagen del captcha sobre el que trabajamos:



Código Completo de la Aplicación

```
<?

#####variables de uso general#####
$width="";
$height="";
$archivos = filelist();
if (empty($archivos)){
    echo "Hola Gracias por dejarme nacer!<br>";
    iniciarsistema();
    echo "<br>Red De Inteligencia activa, 27 caracteres en mi memoria.";
    Exit;
}
$archivos = substr($archivos, 0, strlen($archivos)-1);
$archivos = explode(".", $archivos);
#####CONFIG#####

//echo leer("http://localhost/ocr/ivalidation.php");
echo leer("captcha.jpg");
#####

//FUNCION RECONOCER, ESTA FUNCION ENCUENTRA EL APRECIDO DE LAS LETRAS (ESTA FUNCION E SLA MAS CUSTOMIZABLE DE
ACUERDO AL CAPTCHA)

function leer($imageurl){
    global $width,$height;
    $width = 15;
    $height = 15;
    $img = imagecreatefromjpeg($imageurl);
    imagejpeg($img,"captcha.jpg");
    $img= limpia($img);
    $imgchar1 = imagecreate($width,$height);
    $imgchar2 = imagecreate($width,$height);
    $imgchar3 = imagecreate($width,$height);
    $imgchar4 = imagecreate($width,$height);
    $imgchar5 = imagecreate($width,$height);
    imagecopy($imgchar1, $img, 1, 1, 4, 10, 10, 12);
    imagecopy($imgchar2, $img, 1, 1, 13, 10, 10, 12);
    imagecopy($imgchar3, $img, 1, 1, 22, 10, 10, 12);
    imagecopy($imgchar4, $img, 1, 1, 31, 10, 10, 12);
```

```

imagecopy($imgchar5, $img, 1, 1, 40, 10, 10, 12);
imagejpeg($imgchar1, "chr1.jpg");
imagejpeg($imgchar2, "chr2.jpg");
imagejpeg($imgchar3, "chr3.jpg");
imagejpeg($imgchar4, "chr4.jpg");
imagejpeg($imgchar5, "chr5.jpg");
$ch1 = crearimagenfile("chr1.jpg");
$ch2 = crearimagenfile("chr2.jpg");
$ch3 = crearimagenfile("chr3.jpg");
$ch4 = crearimagenfile("chr4.jpg");
$ch5 = crearimagenfile("chr5.jpg");
$datamap1 = crearchardatamap($ch1, "noname");
$datamap2 = crearchardatamap($ch2, "noname");
$datamap3 = crearchardatamap($ch3, "noname");
$datamap4 = crearchardatamap($ch4, "noname");
$datamap5 = crearchardatamap($ch5, "noname");
$datamap1 = explode("@", $datamap1);
$datamap1 = substr($datamap1[1], 1, strlen($datamap1[1]));
$datamap1 = explode(":", $datamap1);
$nummap1 = count($datamap1);
$datamap2 = explode("@", $datamap2);
$datamap2 = substr($datamap2[1], 1, strlen($datamap2[1]));
$datamap2 = explode(":", $datamap2);
$nummap2 = count($datamap2);
$datamap3 = explode("@", $datamap3);
$datamap3 = substr($datamap3[1], 1, strlen($datamap3[1]));
$datamap3 = explode(":", $datamap3);
$nummap3 = count($datamap3);
$datamap4 = explode("@", $datamap4);
$datamap4 = substr($datamap4[1], 1, strlen($datamap4[1]));
$datamap4 = explode(":", $datamap4);
$nummap4 = count($datamap4);
$datamap5 = explode("@", $datamap5);
$datamap5 = substr($datamap5[1], 1, strlen($datamap5[1]));
$datamap5 = explode(":", $datamap5);
$nummap5 = count($datamap5);
#####CODIGO POR LETRA CAPTCHA#####
$texto .= charletter($datamap1, "chr1.jpg");
$texto .= charletter($datamap2, "chr2.jpg");
$texto .= charletter($datamap3, "chr3.jpg");
$texto .= charletter($datamap4, "chr4.jpg");
$texto .= charletter($datamap5, "chr5.jpg");
unlink("chr1.jpg");
unlink("chr2.jpg");
unlink("chr3.jpg");
unlink("chr4.jpg");
unlink("chr5.jpg");
#####
return "Imagen Captcha <img src='captcha.jpg'><br>Texto Reconocido $texto";
#####
}

```

//ESTA FUNCION DETECTA LETRAS

```

function charletter($datamap1, $file){
    global $archivos;
    $errores = 0;
    for ($i=0; $i<count($archivos); $i++){
        $data = base64_decode(str_rot13(file_read($archivos[$i])));
        $data = explode("@", $data);
        $data = substr($data[1], 1, strlen($data[1]));
        $data = explode(":", $data);
        $numdata = count($data);
        for ($x=0; $x<=$numdata-1; $x++){
            if (($data[$x]) != ($datamap1[$x])){$errores++;}
        }
        $errores[$i] = $errores;
    }
}

```

```

        $erroresb[$i] = $archivos[$i];
        $errores = 0;
    }
    $value = min($erroresa);
    $key = array_search($value, $erroresa);
    $letra = base64_decode(str_rot13(file_read($erroresb[$key])));
    $letra = explode("@", $letra);
    if ($value != 0){learnchar($file, $letra[0]);}
    return $letra[0];
}
//ESTA FUNION LEE LOS ARCHIVOS
function file_read($filename){
    $filename= "ocr/".trim($filename);
    $handle = fopen($filename, "r");
    $contents = fread($handle, filesize($filename));
    fclose($handle);
    return $contents;
}
//ESTA FUNCION DEVUELVE EN UNA VARIABLE LOS ARCHIVOS CONTENIDOS EN EL DIRECTORIO
function file_list(){
    $archivos="";
    $dir = opendir("ocr/");
    while (false !== ($file = readdir($dir))){
        if(strpos($file, "charmap")){$archivos .= $file.",";}
    }
    return $archivos;
}
//ESTA FUNCION LE ENSEÑA AL PROGRAMA A RECONOCER EL ABECEDARIO BASICO
function iniciarsistema(){
    learnchar("muestreo/A.JPG", "A");
    learnchar("muestreo/B.JPG", "B");
    learnchar("muestreo/C.JPG", "C");
    learnchar("muestreo/D.JPG", "D");
    learnchar("muestreo/E.JPG", "E");
    learnchar("muestreo/F.JPG", "F");
    learnchar("muestreo/G.JPG", "G");
    learnchar("muestreo/H.JPG", "H");
    learnchar("muestreo/I.JPG", "I");
    learnchar("muestreo/J.JPG", "J");
    learnchar("muestreo/K.JPG", "K");
    learnchar("muestreo/L.JPG", "L");
    learnchar("muestreo/M.JPG", "M");
    learnchar("muestreo/N.JPG", "N");
    learnchar("muestreo/O.JPG", "O");
    learnchar("muestreo/P.JPG", "P");
    learnchar("muestreo/Q.JPG", "Q");
    learnchar("muestreo/R.JPG", "R");
    learnchar("muestreo/S.JPG", "S");
    learnchar("muestreo/T.JPG", "T");
    learnchar("muestreo/U.JPG", "U");
    learnchar("muestreo/V.JPG", "V");
    learnchar("muestreo/W.JPG", "W");
    learnchar("muestreo/X.JPG", "X");
    learnchar("muestreo/Y.JPG", "Y");
    learnchar("muestreo/Z.JPG", "Z");
    unlink("muestreo/A.JPG");
    unlink("muestreo/B.JPG");
    unlink("muestreo/C.JPG");
    unlink("muestreo/D.JPG");
    unlink("muestreo/E.JPG");
    unlink("muestreo/F.JPG");
    unlink("muestreo/G.JPG");
    unlink("muestreo/H.JPG");
    unlink("muestreo/I.JPG");
    unlink("muestreo/J.JPG");
    unlink("muestreo/K.JPG");
}

```

```

unlink("muestreo/L.JPG");
unlink("muestreo/M.JPG");
unlink("muestreo/N.JPG");
unlink("muestreo/O.JPG");
unlink("muestreo/P.JPG");
unlink("muestreo/Q.JPG");
unlink("muestreo/R.JPG");
unlink("muestreo/S.JPG");
unlink("muestreo/T.JPG");
unlink("muestreo/U.JPG");
unlink("muestreo/V.JPG");
unlink("muestreo/W.JPG");
unlink("muestreo/X.JPG");
unlink("muestreo/Y.JPG");
unlink("muestreo/Z.JPG");
rmdir("muestreo");
}
//ESTA FUNCION LIMPIA EL BACKGROUND DE LAS IMAGENES TRATANDO DE DEJARLA EN DOS COLORES B/N
function limpia ($image){
    $w = imagesx($image);
    $h = imagesy($image);
    for ($x = 0; $x <= $w; $x++){
        for ($i = 0; $i <= $h; $i++){
            $rgb = imagecolorat($image, $x, $i);
            $r = ($rgb >> 16) & 255;
            $g = ($rgb >> 8) & 255;
            $b = $rgb & 255;
            if (($r > 55) || ($g > 55) || ($b > 55)){
                imagesetpixel($image, $x, $i, imagecolorallocate($image, 255, 255, 255));
            }else{
                imagesetpixel($image, $x, $i, imagecolorallocate($image, 0, 0, 0));
            }
        }
    }
    return $image;
}
//funcion para aprender
function learnchar($image,$name){
    global $width,$height;
    $imagen = crearimagenfile($image);
    $datamap = str_rot13(base64_encode(crearchardatamap($imagen,$name)));
    guardar($datamap,$name);
    echo "Nueva relacion de memoria creada, aprendiendo de $name <br>";
}
//CREA IMAGEN TEMPORAL
function crearimagenfile($img_file){
    global $width,$height;
    $img=0;
    $img_sz = getimagesize( $img_file );
    switch( $img_sz[2] ){
        case 1:
            $img = ImageCreateFromGif($img_file);
            break;
        case 2:
            $img = ImageCreateFromJpeg($img_file);
            break;
        case 3:
            $img = ImageCreateFromPng($img_file);
            break;
    }
    $width = $img_sz[0];
    $height = $img_sz[1];
    return $img;
}
//CREA LOS CARACTERES
function crearchardatamap($image,$name){

```

```

        global $width,$height;
        $datamap=$name."@";
        for ($x=0; $x<=$width; $x++){
            for ($y=0; $y<=$height; $y++){
                $datamap.=".$x.".$y."."pixelcolor($image,$x,$y);
            }
        }
        return $datamap;
    }
}
//OBTIENE EL COLOR DE LOS PÍXELES DEFINIDOS
function pixelcolor($im, $x, $y){
    $rgb = imagecolorat($im,$x,$y);
    $r = ($rgb >> 16) & 255;
    $g = ($rgb >> 8) & 255;
    $b = $rgb & 255;
    if (($r > 55) || ($g > 55) || ($b > 55)){
        return 0;
    }
    return 1;
}

}

//SALVAR CHAR DATAMAP

function guardar($datamap,$name){
    $fl = fopen("ocr/$name-".md5(time()*rand(1,100))."-charmap.datamap","w+");
    fwrite($fl,$datamap);
    fclose($fl);
}

}

?>

```

Conclusión

Como vemos crear una aplicación de reconocimiento de imágenes totalmente hecha en casa no es tan complicado como parecía, una vez que lo hacemos comprendemos el manejo de las imágenes y sus formatos, y porque otras técnicas publicadas en la red son un mito.

Como siempre espero que les haya gustado mi manual, y le doy las gracias a toda la banda, en especial a Megabyte porque el lo soñó, pero yo lo hice realidad.

Luis Alberto Cortes Zavala – NaPa

IT / Security Consultant

Security Nation Labs – México

<http://www.securitynation.com>

napa@securitynation.com