Reporte de Implementación - Rompimiento de CAPTCHA con Aprendizaje de Máquina

Didier Muñoz Díaz
Ricardo Figueroa Riestra
Aprendizaje de Máquina
ITAM, 2017

ÍNDICE GENERAL

CAPÍTULO 1

INTRODUCCIÓN

1.1. Introducción

Este proyecto consiste en utilizar técnicas de aprendizaje de máquina para romper[1] un CAPTCHA.
Con base a lo mencionado por Blum *et al.* [1], CAPTCHA (Completlely Automated Public Tests to tell Computers and Humans Apart) es una prueba automatizada que los humanos pueden pasar con facilidad pero que presenta cierta dificultad para una computadora. En su forma original un CAPTCHA es un generador de imágenes de caracteres o palabras en las que a través de distorsiones, transformaciones, ruido o diversas otras técnicas se busca ofuscar el contenido para dificultarle a una computadora distinguirlo o clasificarlo.
Los CAPTCHAs han ido evolucionando con el tiempo conforme se han ido rompiendo. Actualmente también son llamados HIPs (Human Interface Proofs) y los modelos más recientes invocan características del ser humano que siguen siendo difíciles o no se han podido simular correctamente por computadora[2].

Actualmente existen técnicas de aprendizaje de máquina y visión computacional que pueden ayudar a las computadoras a aprender a resolver estos problemas.

Este documento presenta la implementación de 3 algoritmos de aprendizaje de máquina y un algoritmo de visión computacional (como alternativas distintas) para el rompimiento de un CAPTCHA de complejidad baja pero que a pesar de eso sigue siendo similar a los utilizados en sitios web desactualizados. El algoritmo de visión computacional se utilizó para separar caracteres de los CAPTCHAs aunque también se utilizaron los CAPTCHAs completos en los algoritmos de aprendizaje de máquina.

[1] *Romper* un CAPTCHA se refiere a lograr que una computadora logre un alto grado de exactitud al resolver un tipo de CAPTCHA particular.

[2] En 2012 se propuso un CAPTCHA que requería el desarrollo de empatía por parte del usuario[4]. En 2015 se publicaron artículos sobre sistemas que lograban romper en cierta medida este CAPTCHA [5].

La metodología propuesta es la siguiente:

1. **Separación de caracteres de CAPTCHA**. Como primer paso se propone utilizar un algoritmo de separación de caracteres mediante la detección de contornos de letras y encapsulamiento en cajas. En particular el algoritmo utilizado es el mostrado en [6], a través de la biblioteca OpenCV.

2. **Aprendizaje de máquina para reconocimiento de imágenes**. Una vez separados los caracteres en regiones, se aplican 3 técnicas de aprendizaje de máquina para identificar de qué carácter se trata.

    - Red neuronal convolucional para la clasificación de letras separadas.
    - Bosque aleatorio para la clasificación de letras separadas.
    - Arboles de decisión (para CAPTCHAs completos y CAPTCHAs con letras separadas separadas).

1.1.1. Redes neuronales

Las redes neuronales son una técnica muy utilizada en el reconocimiento de patrones por su habilidad de extraer en capas las distintas características presentes en los datos. Las arquitecturas más utilizadas para el procesamiento de imágenes son las llamadas redes neuronales convolucionales de las que las partes más importantes de este tipo de arquitecturas son:

- La capa de entrada es la capa donde se lee directamente la imagen. Suele constar de varios filtros en paralelo del tipo convolucional (convolucionan regiones de la imagen) para obtener un mapa de características. La extracción de información por regiones permite aprender sobre los rasgos de la imagen.

- Etapa de rectificación: Por medio de neuronas rectificadoras se regula el gradiente de aprendizaje y se mejora el tiempo de cómputo. Este tipo de neuronas son biológicamente plausibles.

- Capas de agrupación (pooling layers). Las capas de agrupado resumen la información encontrada en neuronas cercanas en la estructura de la red para reducir la complejidad mientras se extraen las características más relevantes. Básicamente son etapas donde los mapas de características se resumen para facilitar el aprendizaje de conceptos más abstractos.

- Posteriormente la salida de las capas anteriores se conecta a una red neuronal completamente conectada o densa donde se terminan de entender las características de mayor abstracción sobre los datos.

El modelo no tiene por qué tener sólo una etapa de convolución y una de agrupamiento, lo usual es que se alternen varias capas de convolución y agrupamiento antes de la red neuronal final.
Entrenar una red neuronal puede ser complicado porque es un modelo con muchos parámetros por ajustar. Aún así algo se puede considerar: como la arquitectura de una red neuronal convolucional controla la cantidad de información a procesar y suele ir disminuyendo con la profundidad, la elección de la arquitectura es importante basarla en la cantidad de información y la capacidad esperada del modelo. El entrenamiento de una red neuronal de estas características utiliza el método de *Back Propagation*, donde el ajuste de los pesos es realizado en la dirección de la salida de la red hacia la entrada de la red. En este trabajo se terminaron generando más de 20000 imágenes (en el caso del CAPTCHA completo) para asegurar el correcto entrenamiento de la red neuronal.
Uno de los modelos de redes neuronales convolucionales más utilizados es el llamado *LeNet-5*. Esta arquitectura que se ha estudiado bastante consiste en lo siguiente:

1. La primer capa convolucional obtiene mapas de características (por medio de filtros) por medio de convoluciones a regiones de la imagen de entrada.

2. La segunda capa de agrupamiento reduce las dimensiones de los mapas de características.

3. La tercera capa es convolucional y vuelve a generar nuevos mapas de características de menores dimensiones.

4. La cuarta capa es de agrupamiento y reduce los mapas de características a datos factibles de ser introducidos a la última etapa.

5. La quinta etapa es una red neuronal densa o completamente conectada.

CAPÍTULO 2

METODOLOGÍA

2.1. Obtención de los datos de entrenamiento y prueba

Los datos para la implementación del programa se obtuvieron de [2]. Dicho repositorio contiene un *set* de imágenes asociado a otro *set* de etiquetas en formato .txt. El set completo contiene 199 datos, cada CAPTCHA contiene 6 letras como en el siguiente ejemplo:

Figura 2.1: Ejemplo de CAPTCHA

2.2. Implementación del algoritmo de separación de caracteres de CAPTCHA

Debido a la reducida cantidad de observaciones en el *set* de entrenamiento y prueba, el problema se intentó resolver a través de la separación de las letras del CAPTCHA (y etiquetas) y posteriormente la utilización de una herramienta de aprendizaje de máquina.

- Definición de función *EncontrarContornos*: Se definió una función (utilizando la librería OpenCV) que mapea los contornos de una imagen y las encapsula en una figura (puede ser un círculo o un rectángulo). La problemática principal de este método es que si dos o más letras se encuentran unidas en algún punto, el algoritmo de visión computacional puede llegar a juntarlas en una sola caja, para ello, dentro de esta función se incluyó un método que revisa las dimensiones de las cajas obtenidas e iterativamente las parte si superan ciertas dimensiones. Si el algoritmo

junta 4 letras, se parte la caja en 4, si el algoritmo junta dos letras, parte la caja en 3, etc. Más adelante se reportan los resultados de la efectividad de este algoritmo en la sección de *Resultados*. Las dimensiones seleccionadas para la separación de letras son las siguientes

- 35 para una sola letra.
- 70 para dos letras.
- Mayor a 70 para tres o más letras. La mayor cantidad de letras *pegadas* fueron 3.

Consideramos que esta función es bastante *artesanal* en la implementación y que se podría automatizar en mayor medida pero dejamos esto como trabajo futuro.

- **Limpieza de los datos efectivamente *separados***: Se obtuvieron los caracteres o letras de los CAPTCHAS que efectivamente fueron separados y se re-escalaron para que todas las observaciones tuvieran las mismas dimensiones (28x28). La dimensión del color de los datos se modificó a color binario (negro y blanco).

- **Aplicación de ruido y replicación de imágenes**: Para resolver el problema de la falta de imágenes suficientes en el entrenamiento de una herramienta de aprendizaje de máquina, tanto las imágenes de CAPTCHA como las etiquetas se replicaron 100 veces (en el caso de separación de letras). Posteriormente, se aplicó ruido a los datos replicados (con las funciones *Gaussian Blur* y *Rotate* del paquete *Image*. Posteriormente el orden de estas replicaciones se cambió de forma aleatoria. Se hizo el mismo procedimiento para el entrenamiento de CAPTCHAs completos pero con un mayor grado de replicación (20,000 imágenes de CAPTCHA completos).

- **Separación de los datos en entrenamiento-prueba**: Se separaron los datos completos y replicados de imágenes y etiquetas en un 75 % de entrenamiento y 25 % de prueba (esto aplica para cada uno de los métodos de separación y CAPTCHA completo).

- Posteriormente se definieron los modelos de aprendizaje de máquina:

  1. Red Neuronal Convolucional: Las redes convolucionales son conocidas por su habilidad para reconocer patrones en las imágenes. La arquitectura de la red neuronal convolucional se tomó basándose en la arquitectura LeNet-5 y en las características de las imágenes a procesar.
     a) Las imágenes de entrada tienen dimensiones de 28x28.
     b) La capa convolucional 1, genera 32 filtros por medio de kernel 5x5.
     c) La capa de *pooling* 1, agrupa por regiones de 2x2, desfasadas en 2.
     d) La capa convolucional 2, genera 64 mapas de características, por medio de un tamaño de kernel de 5x5.

e) La capa de *pooling* 2, vuelve a agrupar por regiones de 2x2, desfasadas en 2.

f) La capa densa utiliza neuronas de tipo ReLu, y una técnica de *dropout*[1]. al 1 % para evitar que el algoritmo se vuelva ineficiente

g) Finalmente una capa de *logits* termina de convertir la salida de la red neuronal a un formato adecuado para la clasificación. En esta capa se utiliza una función *Softmax* debido a su continuidad.

Se seleccionó una función de costo *Cross Entropy* por su habilidad para realizar clasificación. Adicionalmente los parámetros que se seleccionaron para esta red convolucional están en línea con las dimensiones de las imágenes que se tratan de clasificar (28 x 28 de dimensión, porque consideramos que es un tamaño adecuado a la cantidad de datos replicados). Inicialmente se seleccionaron dimensiones mayores pero se tuvieron problemas con la memoria de la computadora

Los filtros seleccionados tienen una dimensión de 5x5 para ambas capas de la red. El tamaño del *pool* intenta reducir a la mitad los datos por capa. La profundidad es de 18 *letras*, esto debido a que el *data set* efectivamente *separado* únicamente incluía este número de letras (posteriormente se reporta en la sección *Resultados* la efectividad del algoritmo computacional para separar las letras). La tasa de aprendizaje que se seleccionó fue del 10 %. Los datos se entrenaron durante 3 horas en observaciones individuales (*epoch* igual a uno) y tamaño de *batch* de 50 observaciones, esto para intentar reducir el tiempo de entrenamiento.

2. Árbol de decisión: Los mismos datos de entrenamiento y prueba se aplicaron en un modelo de arboles de decisión. Este modelo también se aplicó para la clasificación de CAPTCHAs completos (sin separar las letras)
3. Bosques Aleatorios: De igual forma, los mismos datos de entrenamiento y prueba se aplicaron en un modelo de Bosques Aleatorios con 100 árboles, esto para obtener un modelo suficientemente robusto.

[1] Dropout es una técnica de ajuste de pesos para una red neuronal donde ocasionalmente no se consideran neuronas durante la propagación del error. Esto tiene la ventaja de que se mejora el entrenamiento de las capas menos profundas mientras se incrementa el rendimiento de cómputo.

CAPÍTULO 3

RESULTADOS

3.1. Resultados

3.1.1. Algoritmo de Visión Computacional

El algoritmo de visión computacional logró separar las 6 letras de los CAPTCHAs en un 81 % de las ocasiones, esto debido a las características del CAPTCHA (letras pegadas, ruido, etc), a continuación se presenta una letra separada:

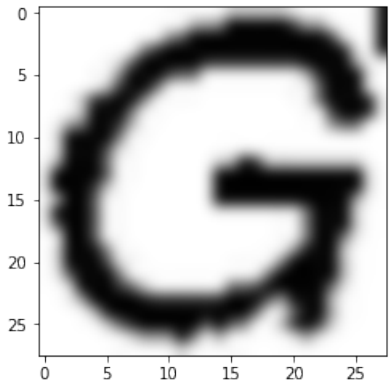

Figura 3.1: Obtención de contornos

En la figura 3.2 se presenta una imagen que muestra la obtención de contornos del algoritmo. En la figura 3.3 se encuentra una imagen que no pudo ser separada totalmente del CAPTCHA.

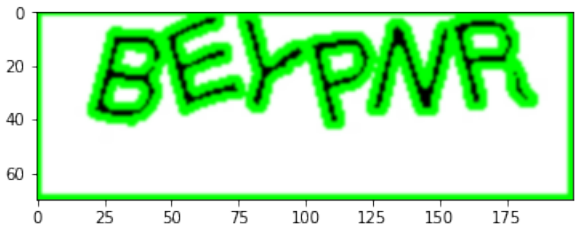

Figura 3.2: Ejemplo de CAPTCHA separado

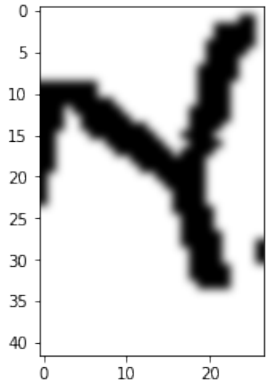

Figura 3.3: Ejemplo de CAPTCHA con separación deficiente

Consideramos que es importante separar la efectividad de esta técnica de la evaluación de los algoritmos de aprendizaje de máquina reportados posteriormente porque las técnicas se pueden aplicar tanto para CAPTCHAs separados como completos. Aunque se implementa un árbol de decisión para clasificar CAPTCHAs completos, consideramos que esta técnica puede ser útil para romper pruebas con mayor complejidad (mayor ruido, tamaños de letra con mucha variación, etc.)

3.1.2. Red Neuronal Convolucional

El rendimiento de la red neuronal se muestra a continuación:

```
INFO:tensorflow:Starting evaluation at 2017-12-08-15:52:36
INFO:tensorflow:Restoring parameters from /tmp/mnist_convnet_model/model.ckpt-22137
INFO:tensorflow:Finished evaluation at 2017-12-08-15:52:57
INFO:tensorflow:Saving dict for global step 22137: accuracy = 0.999019, global_step =
```

Una red neuronal convolucional de las dimensiones (complejidad) escogidas ha sido una red neuronal que ha dado muy buenos resultados. Una exactitud (accuracy) de 99.9 % y

un valor de la función de pérdida (loss) de 0.001855 indican que el modelo ha aprendido a reconocer las letras prácticamente a la perfección. Este resultado va acorde con el hecho de que investigación sobre esta arquitectura de redes neuronales reporta buen desempeño en reconocimiento de imágenes. Este resultado también muestra la correcta elección de la complejidad de la red. En un momento se pensó en que se podría incurrir en sobreajuste, no fue el caso y en la literatura se describen casos similares en los que el sobreajuste empieza a aparecer hasta después de 10 épocas (en ete trabajo sólo se realizó 1 época).

3.1.3. Árbol de decisión

El reporte de clasificación del árbol de decisión implementado se muestra en la Tabla 3.1: La *precisión* del modelo fue muy alta, del 98 % general. Es decir, este modelo tiene una alta

letra	precision	recall	f1-score	support
0	0.98	1.00	0.99	1222
1	1.00	1.00	1.00	1518
2	0.98	1.00	0.99	1580
3	1.00	1.00	1.00	1258
4	0.93	0.98	0.95	1283
5	1.00	1.00	1.00	1269
6	1.00	0.92	0.96	1362
7	0.98	1.00	0.99	1448
8	1.00	0.98	0.99	1346
9	0.99	1.00	0.99	1590
10	1.00	0.92	0.96	1056
11	0.95	1.00	0.97	1358
12	1.00	0.97	0.98	1582
13	0.98	0.97	0.97	1463
14	0.95	1.00	0.97	1488
15	1.00	0.97	0.99	1184
16	1.00	1.00	1.00	1465
17	1.00	0.98	0.99	1080
avg / total	0.98	0.98	0.98	24552

Tabla 3.1: Reporte de clasificación para el árbol de decisión.

capacidad para no clasificar como positivas las letras que no lo son. Por otro lado, un *recall* de 98 % indica que el modelo es muy capaz de encontrar los positivos durante la clasificación. Los árboles de decisión son modelos que en general logran trabajar bien incluso cuando el modelo original que genera los datos no cumple con los supuestos que el árbol

considera[3]. En este caso, el modelo original que genera los datos se puede considerar el algoritmo de generación de los CAPTCHAs. Este algoritmo probablemente funcione generando una imagen de una palabra aleatoria para posteriormente distorsionarla de distintas formas. Si pensamos que el árbol de decisión busca identificar la palabra y para ello en principio asume que cada clase corresponde a una figura base sin modificar, entonces los supuestos que el modelo original viola corresponden a las distorsiones hechas a las imágenes. Esto explicaría el buen funcionamiento del árbol.
En la Tabla 3.2 se muestra la matriz de confusión del árbol de decisión.

	A	B	C	E	F	G	H	J	K	L	M	N	P	R	T	U	X	Y
A	1222	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	1518	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	1580	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	1258	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	1254	0	0	0	0	0	0	29	0	0	0	0	0	0
G	0	0	0	0	0	1269	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	1255	0	0	23	0	26	0	0	58	0	0	0
J	0	0	0	0	0	0	0	1448	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	1323	0	0	0	0	0	23	0	0	0
L	0	0	0	0	0	0	0	0	0	1590	0	0	0	0	0	0	0	0
M	0	0	30	0	23	0	0	27	0	0	976	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	1358	0	0	0	0	0	0
P	0	0	0	0	53	0	0	0	0	0	0	0	1529	0	0	0	0	0
R	30	0	0	0	0	0	0	0	0	0	0	21	0	1412	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1488	0	0	0
U	0	0	0	0	0	0	0	0	0	0	0	0	0	32	0	1152	0	0
X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1465	0
Y	0	0	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0	1055

Tabla 3.2: Matriz de confusión del árbol de decisión.

De esta matriz se logra identificar el alto grado de exactitud del modelo al ver que el soporte de cada clase se encuentra prácticamente en su totalidad en la diagonal principal. Entre las letras más confundidas se encuentran la H con la T y la P con la F. Con las H y T probablemente se debe a la intersección de líneas perpendiculares. Con la P y la F ambas son la misma figura base y difieren en que la P cierra las dos líneas horizontales. Estos dos casos ilustran cómo el árbol de decisión aprende las características de las figuras.

Para mejorar el resultado obtenido por el árbol de decisión, un enfoque posible es un bosque aleatorio. Si ya se vio que un árbol de decisión funciona bien, entonces un bosque aleatorio debe funcionar mejor.

3.1.4. Bosque aleatorio

Los resultados del bosque aleatorio se muestran a continuación. La Tabla 3.3 contiene el reporte de clasificación, en él se confirma la sospecha del buen rendimiento esperado. *Precisión*=100 % y *recall*=100 % implican que el modelo logró encontrar todos los positivos

y discernir todos los negativos como tales. Esto se ilustra en la matriz de confusión en la Tabla 3.4.

letra	precisión	recall	f1-score	support
0	1.00	1.00	1.00	1222
1	1.00	1.00	1.00	1518
2	1.00	1.00	1.00	1580
3	1.00	1.00	1.00	1258
4	1.00	1.00	1.00	1283
5	1.00	1.00	1.00	1269
6	1.00	1.00	1.00	1362
7	1.00	1.00	1.00	1448
8	1.00	1.00	1.00	1346
9	1.00	1.00	1.00	1590
10	1.00	1.00	1.00	1056
11	1.00	1.00	1.00	1358
12	1.00	1.00	1.00	1582
13	1.00	1.00	1.00	1463
14	1.00	1.00	1.00	1488
15	1.00	1.00	1.00	1184
16	1.00	1.00	1.00	1465
17	1.00	1.00	1.00	1080
avg / total	1.00	1.00	1.00	24552

Tabla 3.3: Reporte de clasificación para el bosque aleatorio.

Una matriz de confusión donde todo el soporte de cada categoría se encuentra en la diagonal principal, implica que el modelo clasificó correctamente el 100 % de los datos.

3.1.5. Árbol de decisión con captchas completos

A modo de comparar el rendimiento de los árboles de decisión sobre los CAPTCHAs completos (sin preprocesarlos previamente para separarlos en letras) se aplicó esta técnica sobre los datos originales y replicados sin separar. En la Tabla 3.5 se muestra el reporte de clasificación de este experimento. Como se alcanza a apreciar, el resultado fue muy favorable consiguiendo una precisión de 100 % y un recall de 100 % también.
En la Tabla 3.6 se ilustra la exactitud del 100 % que tuvo el árbol de decisión. Este resultado fue sorprendente pero no inesperado considerando el rendimiento que tuvo en clasificar los caracteres separados. Este resultado además implica la ventaja de no depender de rendimiento del algoritmo de separación de letras.

	A	B	C	E	F	G	H	J	K	L	M	N	P	R	T	U	X	Y
A	1222	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	0	1518	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C	0	0	1580	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	1258	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	1283	0	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	0	0	0	1269	0	0	0	0	0	0	0	0	0	0	0	0
H	0	0	0	0	0	0	1362	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	1448	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	1346	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	1590	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	1056	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	1358	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	1582	0	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0	0	1463	0	0	0	0
T	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1488	0	0	0
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1184	0	0
X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1465	0
Y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1080

Tabla 3.4: Matriz de confusión para el bosque aleatorio.

	precision	recall	f1-score	support
ABXTXM	1.00	1.00	1.00	22
AFCUFA	1.00	1.00	1.00	24
AGHHLT	1.00	1.00	1.00	27
AHAFGT	1.00	1.00	1.00	28
AJJLKR	1.00	1.00	1.00	26
ALECKG	1.00	1.00	1.00	30
ANENGM	1.00	1.00	1.00	24
AYUTCU	1.00	1.00	1.00	27
BBBUUB	1.00	1.00	1.00	31
BBGEGN	1.00	1.00	1.00	27
BCTNPF	1.00	1.00	1.00	25
BEEAHL	1.00	1.00	1.00	27
BEYPNR	1.00	1.00	1.00	25
		...		
XLCYGP	1.00	1.00	1.00	17
XLTXXE	1.00	1.00	1.00	30
XMGJFJ	1.00	1.00	1.00	32
XNULPP	1.00	1.00	1.00	30
XTTUYJ	1.00	1.00	1.00	34
XXERAP	1.00	1.00	1.00	24
YELURL	1.00	1.00	1.00	29
YJLRLE	1.00	1.00	1.00	19
YKENRX	1.00	1.00	1.00	23
YPLGXN	1.00	1.00	1.00	19
YPTGGX	1.00	1.00	1.00	31
YRCNFA	1.00	1.00	1.00	15
YRHKXR	1.00	1.00	1.00	23
avg / total	1.00	1.00	1.00	5104

Tabla 3.5: Reporte de clasificación para el árbol de decisión con CAPTCHAs completos. Se muestra sobre un soporte del 25 %

	1	2	3	...	19998	19999	20000
1	22	0	0	...	0	0	0
2	0	24	0	...	0	0	0
3	0	0	27	...	0	0	0
...				...			
19998	0	0	0	...	31	0	0
19999	0	0	0	...	0	15	0
20000	0	0	0	...	0	0	23

Tabla 3.6: Matriz de confusión para el árbol de decisión con CAPTCHAs completos.

CAPÍTULO 4

CONCLUSIONES

4.1. Conclusiones y trabajo futuro

Con base en los resultados obtenidos, el modelo que representa la mejor solución al problema presentado en este documento es un *Random Forest* en el caso de letras separadas. EL modelo de árboles de decisión para CAPTCHA completo también es adecuado.
A pesar de que el *set* de imágenes utilizadas en este trabajo no constituye un ejemplo de complejidad avanzada, consideramos que los algoritmos implementados son una buena base para el rompimiento de CAPTCHAs más complejos. Los resultados obtenidos son muy positivos, sin embargo incluimos lo siguiente como trabajo futuro o potenciales mejoras:

- Probar y ajustar los algoritmos implementados con CAPTCHAs de mayor complejidad.
- Presentar una solución más efectiva para la separación de letras (81 % de efectividad).
- Implementar mayor automatización en el algoritmo de separación de caracteres.
- También como trabajo futuro sería bueno implementar un *Magic Loop* para la selección de modelos.
- Implementación de *Cross Validation* para la selección de parámetros de los modelos seleccionados.

BIBLIOGRAFÍA

[1] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In *Proc. IACR Eurocrypt*, pages 294–311, 2003.

[2] *ÏX Captcha Solver,"* Retrieved from `https://www.kaggle.com/c/ix-captcha-solver/dAta`

[3] T. Hastie, R. Tibshirani and J. Friedman, "The Elements of Statistical Learning", Springer series in Statistics, Springer, New York, 2001.

[4] Civil Rights Defenders website, .Our captcha has become world news", [online] In: https://www.civilrightsdefenders.org/news/new-web-tool-detects-human-empathy/

[5] Hernández-Castro C.J., Barrero D.F., R-Moreno M.D. .A Machine Learning Attack against the Civil Rights CAPTCHAÏn: Camacho D., Braubach L., Venticinque S., Badica C. (eds) Intelligent Distributed Computing VIII. Studies in Computational Intelligence, vol 570. Springer, Cham, 2015.

[6] S. Suzuki and K Abe, "Topological Structural Analysis of Digitized Binary Images by Border Following", Computer Vision, Graphics and Image Processing, 1985.