



Uso de Aprendizaje de Máquina para el análisis de los ficheros de la DFS

Asesores:

Mariana Esther Martínez Sánchez

Juan Carlos

(Asesor modelos analíticos)

Estudió Matemáticas Aplicadas y
Computación

Le gusta la robótica y programar

Su lenguaje favorito es Python



► <https://github.com/datadogmx>

► perrosdatos.com

Contenido 1

1. Objetivo
2. Datos históricos de los ficheros de la DFS
3. Obtención y características de los archivos
4. Reconocimiento óptico de caracteres
 - 4.1 Flujo OCR
 - 4.2 Búsqueda de hiper-parámetros
 - 4.3 Primera ejecución
 - 4.4 Segunda ejecución
5. Modelos para mejorar el pipeline
6. Bibliografía

Objetivo del proyecto

Con el uso de herramientas de Aprendizaje de Máquina es posible extraer y reconstruir la información sobre la maquinaria de vigilancia de la DFS la cual se encuentra contenida dentro de los ficheros.

Procesar mediante OCR la totalidad de los ficheros buscando obtener la mayor cantidad posible de datos que permitan su identificación y posterior clasificación.

Contenido 2

1. Objetivo
2. Datos históricos de los ficheros de la DFS
3. Obtención y características de los archivos
4. Reconocimiento óptico de caracteres
 - 4.1 Flujo OCR
 - 4.2 Búsqueda de hiper-parámetros
 - 4.3 Primera ejecución
 - 4.4 Segunda ejecución
5. Modelos para mejorar el pipeline
6. Bibliografía

Datos históricos de los ficheros de la DFS

Los ficheros

- Contiene entre 60 a 80 millones de fichas
- Mencionan entre 3 a 4 millones de personas o instituciones.
- Luis Vargas González elaboró la (primera) nomenclatura en 1924
- Paso al resguardo del Archivo General de la Nación (AGN) en enero de 1982
- Digitalizado durante las investigaciones de la FEMOSPP y la COMVERDAD
- Actualmente parcialmente disponible en linea en Archivos de la Represión (Artículo 19)

Contenido 3

1. Objetivo
2. Datos históricos de los ficheros de la DFS
3. Obtención y características de los archivos
4. Reconocimiento óptico de caracteres
 - 4.1 Flujo OCR
 - 4.2 Búsqueda de hiper-parámetros
 - 4.3 Primera ejecución
 - 4.4 Segunda ejecución
5. Modelos para mejorar el pipeline
6. Bibliografía

Los Ficheros

Obtención de los archivos

En la página <https://archivosdelarepresion.org/> se encuentra la sección Archivos, ahí podemos encontrar un link hacia el Archivo sin catalogar(ver Figura 1).



Figura: Sección Archivo en web:

<https://archivosdelarepresion.org/>

Los Ficheros

Características de los archivos

Después de la descarga de archivos, los 18,008 Ficheros descargados se distribuyen en 9 resoluciones (ver Tabla 1). De ese total el 76 % (13,826) de las imágenes son tomadas horizontalmente.

Altura	Anchura	Conteo	Porcentaje(%)
1728	2304	792	4.40
1944	2592	6	0.03
2304	1728	1876	10.42
2448	3264	722	4.01
2592	1944	2637	14.64
3264	2448	413	2.29
3456	4608	2662	14.78
4320	2432	188	1.04
4608	3456	8712	48.38

Cuadro: Resolución de los Ficheros.

Los Ficheros

Características de los archivos

La profundidad de las imágenes se distribuye en **3,699** imágenes con solo un nivel de profundidad, **12,797** imágenes con solo dos niveles de profundidad y **1,512** con tres niveles de profundidad. La totalidad de los archivos y carpetas en total representan **54.8GB** de información.

Contenido 4

1. Objetivo
2. Datos históricos de los ficheros de la DFS
3. Obtención y características de los archivos
4. Reconocimiento óptico de caracteres
 - 4.1 Flujo OCR
 - 4.2 Búsqueda de hiper-parámetros
 - 4.3 Primera ejecución
 - 4.4 Segunda ejecución
5. Modelos para mejorar el pipeline
6. Bibliografía

Reconocimiento óptico de caracteres

Definición corta

Parafraseando un poco a [3]:

El principal objetivo de los repositorios institucionales y librerías digitales es poner a disposición y facilitar el intercambio de conocimiento es decir proporciona literatura de libre acceso, bajo este enfoque el OCR ayuda como herramienta para escanear documentos físicos y colecciónar la información contenida en estos de forma digital para su posterior explotación.

Reconocimiento óptico de caracteres

Flujo OCR

Para la preparación de las imágenes Tesseract recomienda el siguiente pipeline : [4].



Figura: Flujo de pre-procesamiento sugerido por tesseract

Reconocimiento óptico de caracteres

Flujo OCR

Ejemplo de ficha pasada por el flujo OCR:

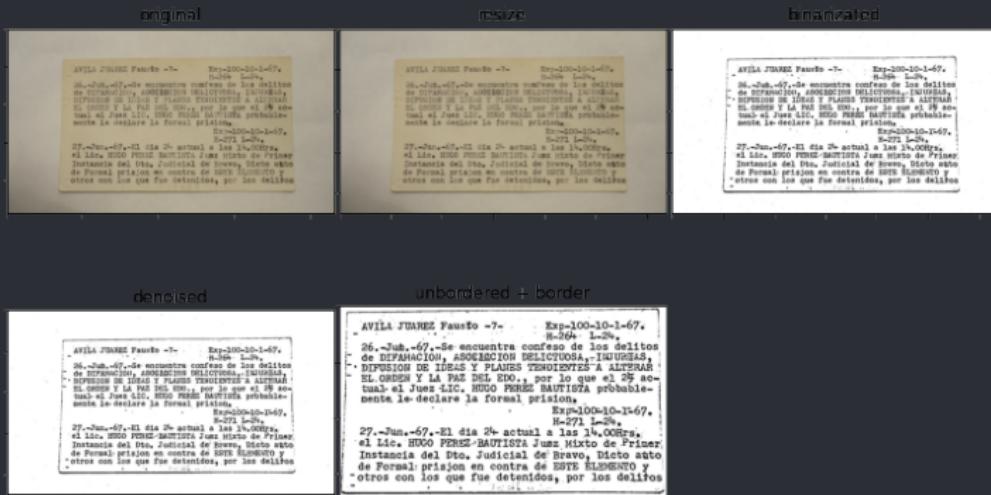


Figura: Flujo de pre-procesamiento aplicado a una ficha del archivo de la represión.

Reconocimiento óptico de caracteres

Búsqueda de hiper-parámetros

En el método propuesto encontramos los siguientes parámetros:

- height: Altura de la imagen resultante en el paso de re-escalamiento.
- border_size: Cantidad de píxeles que dejaremos al borde de la imagen al recortar la ficha.
- sigma: Parámetro que modifica el umbral de detección de bordes.

Reconocimiento óptico de caracteres

Búsqueda de hiper-parámetros

Elegimos una muestra aleatoria de 500 fichas junto con los siguientes posibles valores:

- height: [900, 1100, 1200, 1300, 1700].
- border_size: [15,20,25].
- sigma: [0.33, 0.01].

Esto da como total: $500 * 5 * 3 * 2 = 1,500$ registros a evaluar (cifra muy cercana a los 18,008 fichas descargadas). Para generar la rejilla de evaluación, hicimos uso de la librería scikit-learn (ver 4)

Reconocimiento óptico de caracteres

Búsqueda de hiper-parámetros

Código 2.8: Generando la rejilla de evaluación.

```
from sklearn.model_selection import ParameterGrid
import random
random.seed(1234)
param_grid = {'path':random.sample(imgs_paths,500),
              'sigma': [0.33, 0.01],
              'pixels' : [900, 1100, 1200, 1300, 1700],
              "border": [15,20,25]}
grid = ParameterGrid(param_grid)
```

Figura: Generando la rejilla de evaluación.

Reconocimiento óptico de caracteres

Evaluando el desempeño del OCR buscando expedientes en el texto

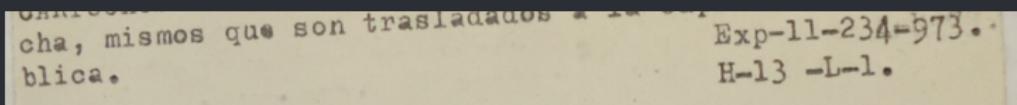
Para **evaluar el desempeño de nuestras lecturas**, la primera métrica que usamos fue buscar expedientes en el texto resultante. Elegimos el expediente por su valor informacional (nos permite reconstruir historias y hacer referencia a un expediente es el principal objetivo de las fichas).

Las lecturas del OCR suelen confundir caracteres, por ejemplo la palabra EXP puede ser leída como 3XP así como los 1 con l, etc. Para tratar de evadir esta limitante hicimos uso de el **RegEx**

Reconocimiento óptico de caracteres

Evaluando el desempeño del OCR buscando expedientes en el texto con RegEx

Los RegEx son secuencias de caracteres que especifican patrones de búsqueda, con el objetivo de extraer datos o remplazarlos siguiendo el patrón como criterio de selección.



(a) El expediente visualmente

trasladados a la capital de la
Ropú-\nblica. Sxp-11-
234=973>-.\\n.. - 4 - > Ha 13

(b) Salida del OCR

Cuadro: Ejemplo de lectura del OCR con errores de caracteres

Reconocimiento óptico de caracteres

Evaluando el desempeño del OCR buscando expedientes en el texto con RegEx

Los RegEx usados son los siguientes:

```
div = "-|_|--|=|__|=|-/-| - "
div_mod = "-|_|--|=|__|=|-/-| / - "
match_regex5 = "((\d{1,3})(div)(\d{1,3})(div_mod)(\d{1,3})(div_mod)(\d{1,3})).replace("div",div).replace("div_mod",div_mod)
match_regex4 = "((\d{1,3})(div)(\d{1,3})(div_mod)(\d{1,3})(div)(\d{1,3})).replace("div",div).replace("div_mod",div_mod)
match_regex3 = "((\d{1,3})(div)(\d{1,3})(div)(\d{1,3})).replace("div",div).replace("div_mod",div_mod)
```

Figura: RegEx definidos para la búsqueda de claves de expediente.

Reconocimiento óptico de caracteres

Búsqueda de hiper-parámetros

En cuanto a los mejores hiper-parámetros encontramos lo siguiente:

altura	ancho	sigma	pixels	border
1728	2304	0.33	1200	15
1944	2592	0.33	1300	15
2304	1728	0.33	1300	20
2432	4320	0.01	900	20
2448	3264	0.33	1200	20
3264	2448	0.33	1300	20
3456	4608	0.01	1300	25
4608	3456	0.33	1200	15

Cuadro: Mejores hiper-parámetros para las resoluciones de los Ficheros.

Reconocimiento óptico de caracteres

Búsqueda de hiper-parámetros

Pareciera ser que bajar a resolución 1200-1300 suele ser la mejor opción para obtener las mejores lecturas. Todo este paso tardó: 5 horas 4 minutos 1 segundo.

Reconocimiento óptico de caracteres

Primera ejecución

Después de obtener los mejores hiper-parámetros para nuestro conjunto de fichas, los resultados del primer procesamiento fueron los siguientes.

Del total de las 18,008 fichas con el primer procesamiento pudimos detectar:

Patrón de expediente detectado	Total
1	9011
0	8997

Cuadro: Patrón de expediente detectados en primer procesamiento.

Reconocimiento óptico de caracteres

Primera ejecución

Poniendo atención a el comportamiento que tuvo el algoritmo por resolución podemos observar que el que la imagen haya sido tomada de forma vertical afecta significativamente la efectividad del algoritmo (ver 5).

ALTURA	ANCHO	HORIZONTAL	F. TOTALES	F. PATRÓN	EFFECTIVIDAD
3456	4608	1	8712	5021	0.58
2448	3264	1	413	244	0.59
2432	4320	1	188	144	0.77
1944	2592	1	2637	1854	0.70
1728	2304	1	1876	1177	0.63
4608	3456	0	2662	333	0.13
3264	2448	0	722	53	0.07
2592	1944	0	6	0	0.00
2304	1728	0	792	185	0.23

Cuadro: Patrón de expediente detectados en primer procesamiento dividido por resolución.

Reconocimiento óptico de caracteres

Primera ejecución

Esto se puede explicar porque al encontrarnos con estos casos usamos el método de segmentación 1 (Automatic page segmentation with OSD).

De las 9,011 fichas detectadas, la mayoría se concentran en el grupo de 3-4 dígitos (ver 6).

Tipo de patrón	Fichas detectadas
Expedientes con 5 dígitos	1584
Expedientes con 4 dígitos	4793
Expedientes con 3 dígitos	9011

Cuadro: Fichas detectadas por tipo de patrón.

Todo el primer procesamiento tardó 6 horas 45 minutos con 44s.

Reconocimiento óptico de caracteres

Segunda ejecución

Después de obtener los resultados del primer procesamiento, algunos de los hallazgos fueron que era una desventaja no tener la información de la confianza de las palabras y su posición en la imagen. Además de que había que buscar un nuevo enfoque en el caso de las imágenes en vertical.

Del total de las 18,008 fichas con el segundo procesamiento pudimos detectar

Patrón de expediente detectado	Total
1	9238
0	8770

Cuadro: Patrón de expediente detectados en segundo procesamiento (1 sí, 0 no).

Reconocimiento óptico de caracteres

Segunda ejecución

La mejora al asumir que girar la ficha 90 grados cuando tenemos una imagen vertical mejoró solo en 227 nuevas fichas con expedientes detectadas.

ALTURA	ANCHO	HORIZONTAL	F. TOTALES	F. PATRÓN	EFECTIVIDAD
3456	4608	1	8712	5024	0.58
2448	3264	1	413	244	0.59
2432	4320	1	188	144	0.77
1944	2592	1	2637	1855	0.70
1728	2304	1	1876	1178	0.63
4608	3456	0	2662	507	0.19
3264	2448	0	722	62	0.09
2592	1944	0	6	0	0.00
2304	1728	0	792	224	0.28

Cuadro: Patrón de expediente detectados en segundo procesamiento dividido por resolución.

Reconocimiento óptico de caracteres

Segunda ejecución

De las 9,238 fichas detectadas, la mayoría se concentran en el grupo de 3-4 dígitos (ver 10).

Tipo de patrón	Fichas detectadas
Expedientes con 5 dígitos	1607
Expedientes con 4 dígitos	4873
Expedientes con 3 dígitos	9238

Cuadro: Fichas detectadas por tipo de patrón en segundo procesamiento.

Todo el segundo procesamiento tardó 7 horas 46 minutos 56 segundos.

Reconocimiento óptico de caracteres

Segunda ejecución: Fuente

En el segundo procesamiento debido a que guardamos los meta-datos del OCR pudimos programar un proceso para detectar el tamaño de la fuente de la ficha al momento de la lectura. El flujo propuesto fue el siguiente:

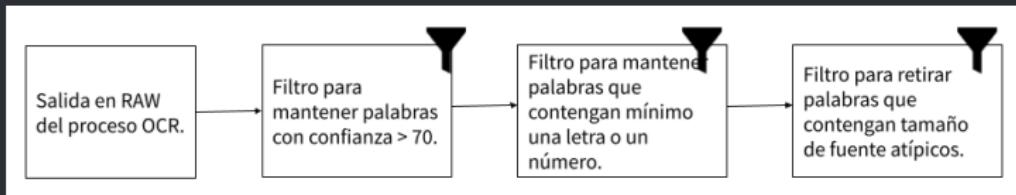


Figura: Flujo para obtener la fuente de las fichas usando la salida del OCR.

Reconocimiento óptico de caracteres

Segunda ejecución: Fuente

Un ejemplo de como se ve utilizado en una ficha:

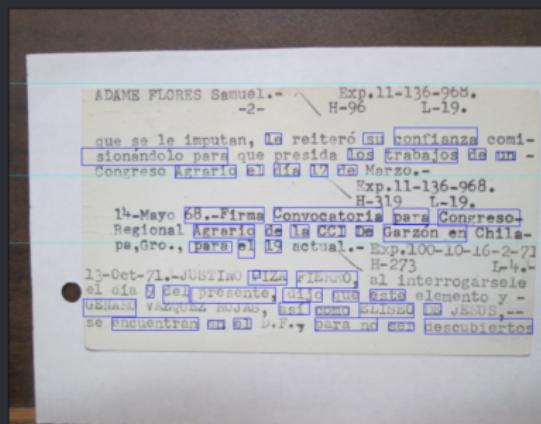
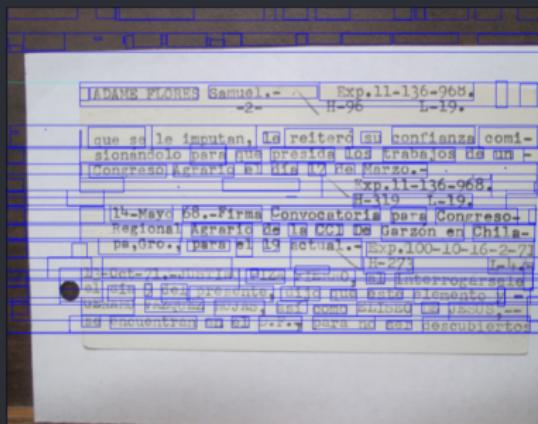


Figura: Ambas imágenes corresponden a la misma ficha, la de lado izquierdo tiene seleccionadas todas las palabras encontradas por el OCR, la de la derecha solo aquellas que cumplen con los filtros para el tamaño de fuente.

Reconocimiento óptico de caracteres

Segunda ejecución: Fuente

El histograma de las fuentes encontradas en ambas fichas:



Figura: Ambos histogramas corresponden a la misma ficha, de lado izquierdo tiene seleccionadas todas las palabras encontradas por el OCR, la de la derecha solo aquellas que cumplen con los filtros.

Reconocimiento óptico de caracteres

Segunda ejecución: Fuente

Detectando el tamaño de fuente de todas las fichas pudimos observar la siguiente distribución.

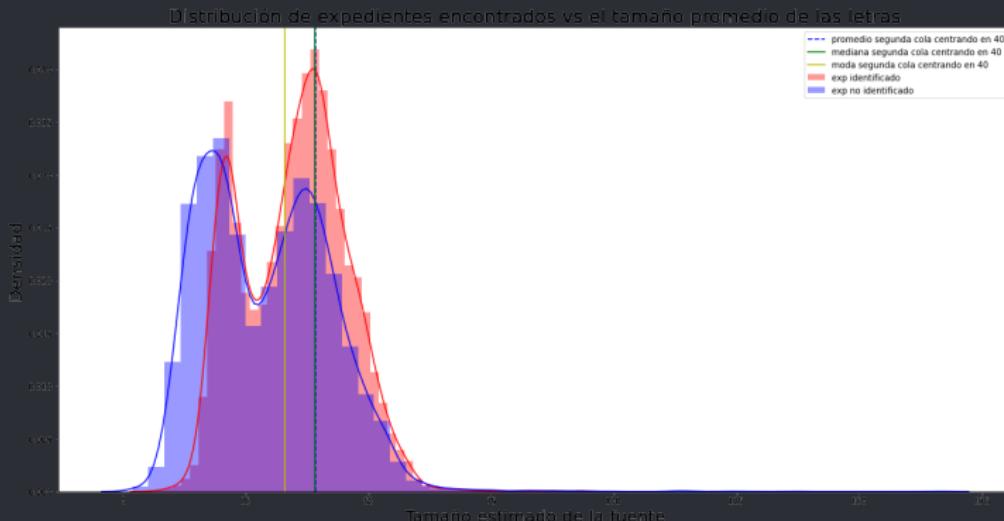


Figura: Distribución del tamaño de la fuente en fichas con patrones de expedientes detectados.

Reconocimiento óptico de caracteres

Segunda ejecución: Fuente

Algo interesante es que al observar los extremos, pudimos visualizar que a veces aglomera archivos que no parecen Fichas.

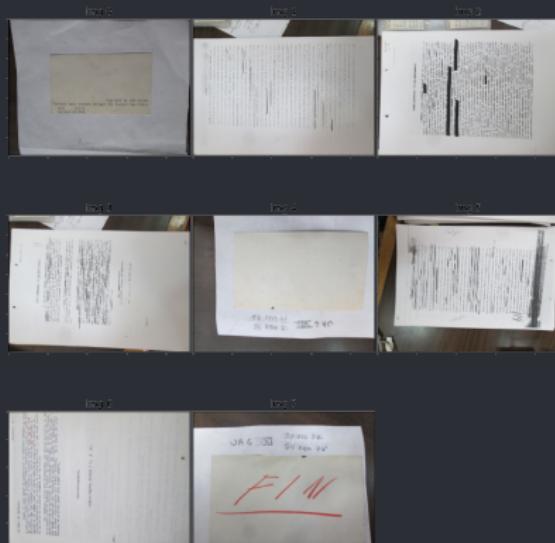


Figura: Muestra de imágenes clasificadas con fuentes menores a 10.

Reconocimiento óptico de caracteres

Segunda ejecución: Fuente

Ejemplo de fichas con fuentes muy grandes.



Figura: Muestra de imágenes clasificadas con fuentes mayores a 100.

Reconocimiento óptico de caracteres

Segunda ejecución: Mas de un expediente por ficha

En cuanto a los expedientes que pudimos detectar en una ficha, encontramos que:

Número de expedientes por ficha	Fichas
8	2
7	10
6	22
5	47
4	166
3	612
2	2552
1	5827

Cuadro: Número de expedientes detectados por Ficha.

Reconocimiento óptico de caracteres

Segunda ejecución: Mas de un expediente por ficha

Entre los casos peculiares se encuentran aquellos casos donde encontramos 8 expedientes en las ficha:

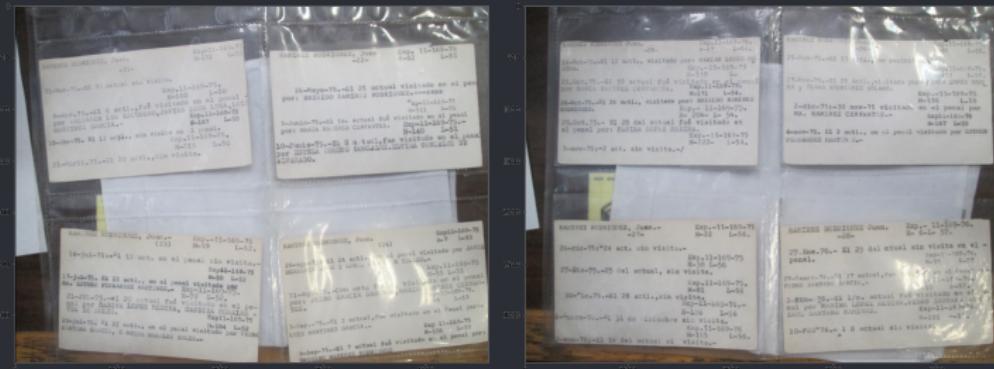


Figura: Las fichas donde detectamos 8 expedientes.

Contenido 5

1. Objetivo
2. Datos históricos de los ficheros de la DFS
3. Obtención y características de los archivos
4. Reconocimiento óptico de caracteres
 - 4.1 Flujo OCR
 - 4.2 Búsqueda de hiper-parámetros
 - 4.3 Primera ejecución
 - 4.4 Segunda ejecución
5. Modelos para mejorar el pipeline
6. Bibliografía

Detector de ficheros

Detector de ficheros:Aplicación web

Uno de los problemas identificados fue que aunque la mayoría de las imágenes dentro de la carpeta Ficheros son ficheros, existían casos donde la fotografía correspondía a otro tipo de documento. Esto impedía programar y evaluar correctamente el desempeño de los algoritmos generados.



Detector de ficheros

Detector de ficheros:Aplicación web

Se pensó en generar un modelo de visión por computadora que nos apoyara en la clasificación de las fotografías, para esto se creo una aplicación web que nos permitiera etiqueta rápidamente un conjunto de fotografías.

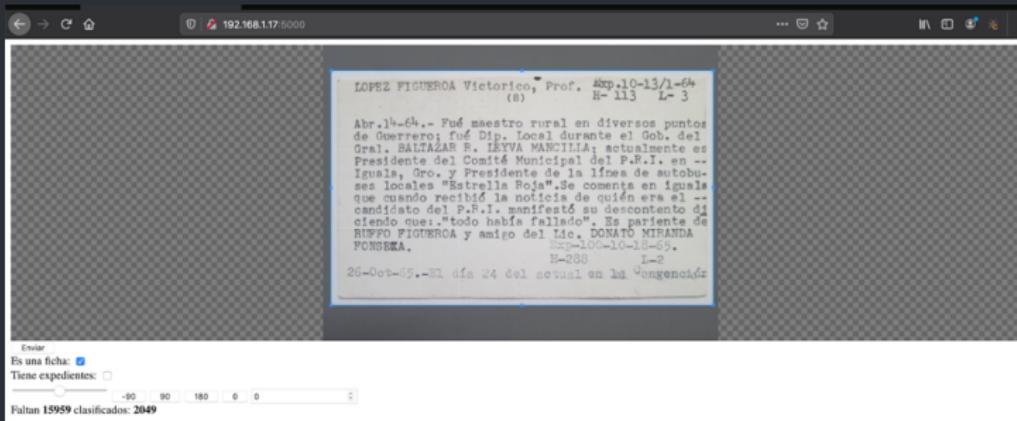


Figura: Aplicación usada para etiquetar un conjunto de las fichas.

Detector de ficheros

Detector de ficheros:Aplicación web

La meta-data disponible para anotar mediante la aplicación fueron las siguientes:

- **is_expediente:** 1 en caso de que la imagen contenga un fichero
0 en caso contrario.
- **has_exp:** bandera identificando si el fichero contiene un expediente anotado.
- **angle:** ángulo necesario para alinear correctamente el texto en la imagen.

Detector de ficheros

Detector de ficheros:Aplicación web

En total se anotaron 2,000 imágenes, se tomaron las imágenes con las que no habíamos podido tomar un expediente mediante el RegEx. Sus características fueron las siguientes:

Es un fichero	Es un expediente	total
0	0	618
	1	1
1	0	640
	1	741

Cuadro: Número de imágenes en cada grupo etiquetado.

Si bien había mas fichas en el conjunto, existía la proporción suficiente para esperar una buena generalización sin necesidad de hacer oversampling.

Detector de ficheros

Detector de ficheros: Data Augmentation

Para poder detectar si la imagen tenia una ficha aunque esta estuviera rotada se uso la plataforma **roboflow**

Se generaron las particiones train-test con 1,000 imágenes para cada grupo, aumentando en la plataforma x3 el conjunto de train, se eligió la transformación rotación dentro de los ángulos 0,90,-90,180.



Figura: roboflow rotaciones generadas.

Detector de ficheros

Detector de ficheros: Entrenamiento

Para entrenar el clasificador se uso la arquitectura **yolov5**, el entrenamiento encajaba con el generado en las anotaciones y mediante el modo transfer-learning nos permitía no comenzar desde cero.

```
train.py -hyp data/hyp.scratch.yaml -img 640 -batch 16 -epochs 10  
-data ficheros.yaml -weights yolov5s.pt
```

Detector de ficheros

Detector de ficheros: Métricas de desempeño

Después de 10 épocas pareciera ser que el modelo aprendió a generalizar e identificaba con un mAP arriba del 95

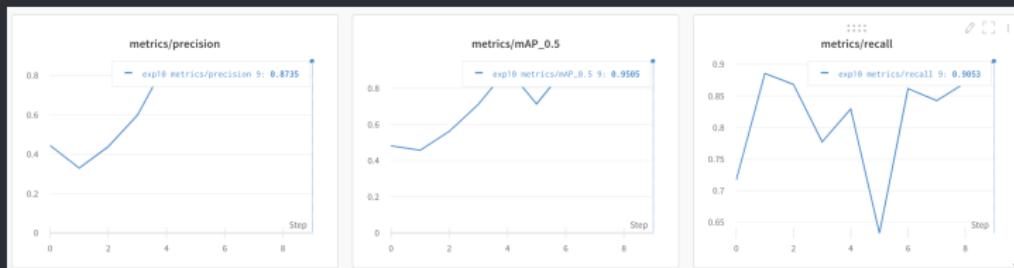


Figura: yolov5 metricas de desempeño (generadas en: [wandb](#)).

Detector de ficheros

Detector de ficheros: Métricas de desempeño

Para tener una mejor métrica del uso que se le dará al modelo, encapsulamos el modelo de pytorch en un modulo de python.

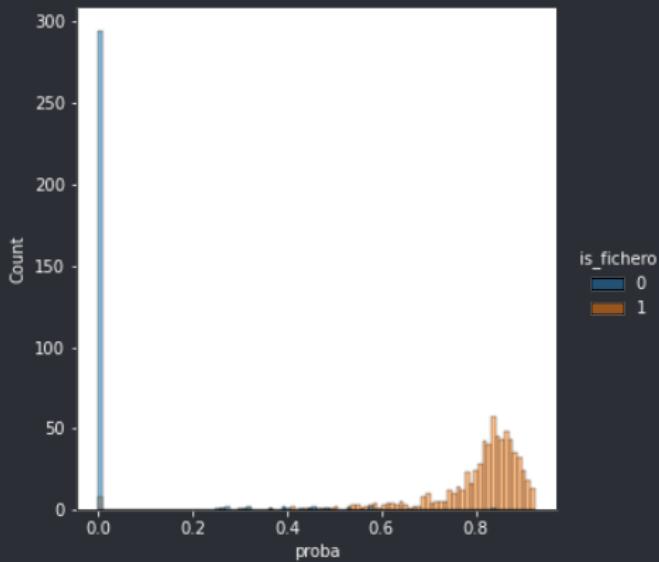


Figura: Evaluando con el conjunto test (imágenes nunca vistas por la red): .

Detector de ficheros

Detector de ficheros: Métricas de desempeño

Las métricas de desempeño para un threshold de 0.4 son las siguientes:

Accuracy : 0.977000

Precisión : 0.983895

Recall : 0.982456

Ganancia en: 0.438442

		Predicción	
		0	1
Actual	0	305	11
	1	12	672

Detector de ficheros

Detector de ficheros: Ejemplo evaluación

Ejemplo visual de una fotografía evaluada.

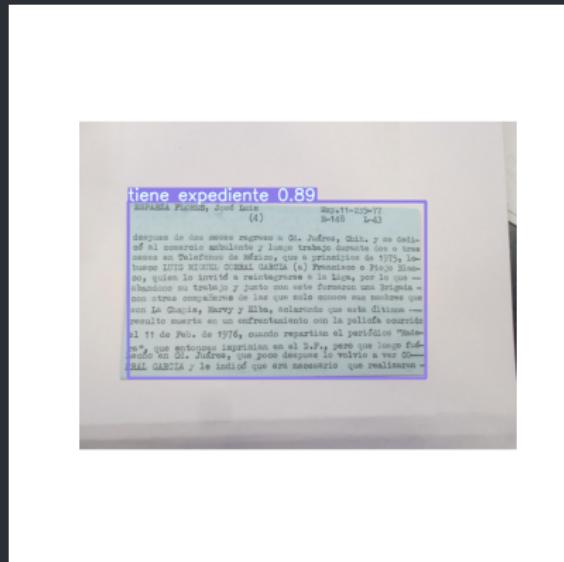
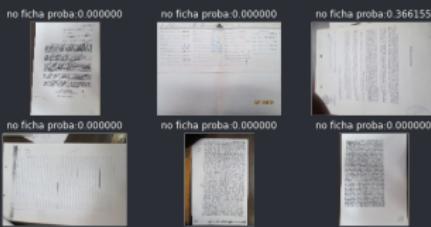


Figura: Fichero evaluado con modelo yolov5 re-entrenado: .

Detector de ficheros

Detector de ficheros: Ejemplo no ficheros

Con el modelo podemos identificar que archivos no son ficheros y con ello apoyar en la labor de identificación en muestras no catalogadas.



Detector de rotación

Una vez habiendo obtenido el área correspondiente a la ficha en sus cuatro ángulos, se evalúa el área con OCR.

Para identificar analíticamente la rotación de la imagen se decidió entrenar un modelo de clasificación binario basado en arboles, es decir un RandomForest. Para entrenar el modelo se uso la muestra de 2,000 mil imágenes, filtrando aquellas que eran ficheros.

Detector de rotación

Variable objetivo

Figura: Definición de variable objetivo del modelo detector de rotación.

Detector de rotación

Distribución de la propensión del modelo propuesto

La implementación del modelo mejora 4 veces la precisión de una decisión aleatoria, pasa de tener 0.2479 a 0.9924.

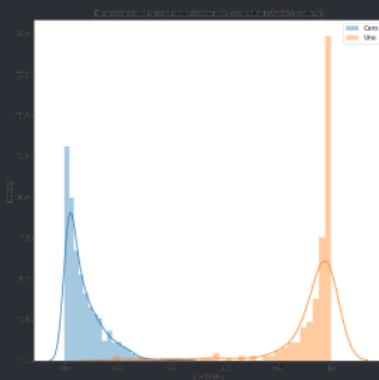


Figura: Desempeño del modelo detector de correcta rotación en el conjunto de prueba.

Detector de rotación

Distribución de la propensión del modelo propuesto

La implementación del modelo mejora 4 veces la precisión de una decisión aleatoria, pasa de tener 0.2479 a 0.9924.

Predicción Actual	0	1	Todos
0	825	0	825
1	9	263	272
Todos	834	263	1097

(a) Matriz de confusión

Métrica	
ROC	0.999361
Recall	0.966912
Precisión	0.992453
Precisión observada(real):	0.247949
Mejora en precisión:	4.002650

(b) Métricas de desempeño del modelo

Cuadro: Métricas de desempeño del modelo detector de rotación

Métrica de calidad de la lectura

Predicted Actual	0	1	All	Métrica
0	695	5	700	Recall
1	187	535	722	Precisión
All	882	540	1422	Precisión observada(real): Mejora en precisión:

(a) Métrica ≥ 0.7 : Matriz confusión

(b) Métrica ≥ 0.7 : métricas desempeño

Predicted Actual	0	1	All	Métrica
0	558	0	558	Recall
1	17	3	20	Precisión
All	575	3	578	Precisión observada(real): Mejora en precisión:

(c) Métrica < 0.7 : Matriz confusión

(d) Métrica < 0.7 : métricas desempeño

Figura: Métricas de desempeño de la métrica de calidad de la lectura propuesta.

Subproblemas encontrados

Después de dos ejecuciones del OCR usando distintas formas de rotar y aplicar el flujo OCR, encontramos los siguientes impedimentos:

- No todas las fotos corresponden a ficheros (**detector de ficheros** actualmente resultó con yolov5 re-entrenado).
- En el caso de los ficheros no es trivial seleccionar el área del texto, las formas tradicionales usan cosas como la tonalidad de las hojas y los supuestos de buenas condiciones como iluminación y fondo. **Seleccionar el área de la imagen que corresponde al texto** ayuda a tener mejores lecturas de las imágenes. Actualmente lo resolvemos con el Yolov5.

Subproblemas encontrados

- La **rotación de la imagen** afecta significativamente la lectura de los ficheros, actualmente lidiamos con este problema usando un modelo de clasificación binario que decide que texto corresponde al mejor ángulo.
- Una **medida de la calidad del texto** es útil, nos ayuda a conocer la confiabilidad de la lectura y optimizar esfuerzos, actualmente usamos un modelo de clasificación binario entrenado con la salida de fichas correctamente rotadas (las fichas que fueron leídas en el ángulo correcto suelen tener palabras ilegibles).
- La **extracción** de las **entidades** y los **expedientes** será útil para el proyecto archivosdelarepresión, actualmente lo atacamos con el uso de RegEx.

Contenido 6

1. Objetivo
2. Datos históricos de los ficheros de la DFS
3. Obtención y características de los archivos
4. Reconocimiento óptico de caracteres
 - 4.1 Flujo OCR
 - 4.2 Búsqueda de hiper-parámetros
 - 4.3 Primera ejecución
 - 4.4 Segunda ejecución
5. Modelos para mejorar el pipeline
6. Bibliografía

Bibliografía

- [1] Artículo19 y COMVERDAD.Archivos de la represión.
2020.url:<https://archivosdelarepresion.org/>.
- [2] INEHRM DEH-INAH. Guía del Fondo de la Secretaría de Gober-nación
Sección: Dirección General de Investigaciones Políticas y So-
ciales.url:<https://www.estudioshistoricos.inah.gob.mx/guia/intro.html>: :text=
- [3] Amarjot Singh, Ketan Bacchuwar, and Akshay Bhasin. “A Sur-vey of
OCR Applications.” In:International Journal of MachineLearning and
Computing.3rd ser.2(2012), pp.314–318.doi:10.7763/ijmlc.2012.v2.137.

Bibliografía

- [4] Improving the quality of the output.url:<https://tesseract-ocr.github.io/tessdoc/ImproveQuality>.
- [5] Sergio Aguayo Quezada.La charola: una historia de los servicios de inteligencia en México. Editorial Ink,2014.