
# Construcción de herramientas para contribuir al análisis de los archivos de la O.C.O.A.

Mateo Nogueira

Programa de Posgrado en Ciencia de Datos y Aprendizaje Automático
Facultad de Ingeniería
Universidad de la República

Montevideo – Uruguay
Noviembre de 2023


UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

# Construcción de herramientas para contribuir al análisis de los archivos de la O.C.O.A.

Mateo Nogueira

Tesis de Maestría presentada al Programa de Posgrado en Ciencia de Datos y Aprendizaje Automático, Facultad de Ingeniería de la Universidad de la República, como parte de los requisitos necesarios para la obtención del título de Magíster en Ciencia de Datos y Aprendizaje Automático.

Directores:
Dr. Gregory Randall
Dra. Lorena Etcheverry

Directora académica:
Dra. Lorena Etcheverry

Montevideo – Uruguay
Noviembre de 2023

Nogueira, Mateo

Construcción de herramientas para contribuir al análisis de los archivos de la O.C.O.A. / Mateo Nogueira. - Montevideo: Universidad de la República, Facultad de Ingeniería, 2023.

x, 148 p. 29, 7cm.

Directores:

Gregory Randall

Lorena Etcheverry

Director académico:

Lorena Etcheverry

Tesis de Maestría – Universidad de la República, Programa en Ciencia de Datos y Aprendizaje Automático, 2023.

Referencias bibliográficas: p. 142 – 142.

1. Análisis de documentos históricos, 2. Reconocimiento de patrones, 3. OCR. I. Randall, Gregory, Etcheverry, Lorena, . II. Universidad de la República, Programa de Posgrado en Ciencia de Datos y Aprendizaje Automático. III. Título.

INTEGRANTES DEL TRIBUNAL DE DEFENSA DE TESIS

Alicia Fernández

Guillermo Moncecchi

Betania Núñez

Ignacio Ramírez

Montevideo – Uruguay
Noviembre de 2023

# Agradecimientos

Quisiera aprovechar este espacio para expresar mi agradecimiento a todas las personas que contribuyeron de manera significativa en la realización de esta tesis. Su apoyo, orientación y dedicación fueron fundamentales para llevar a cabo este trabajo de investigación.

En primer lugar, deseo expresar mi gratitud a mis tutores, Gregory y Lorena, por su guía y supervisión a lo largo de todo el proceso. Su experiencia, conocimientos y paciencia fueron fundamentales para el desarrollo de este proyecto.

También quiero agradecer de manera especial a Guillermo, quien me brindó la oportunidad de visitar la Universidad de Duke. Durante esos días, pude realizar los avances más significativos de toda la investigación. Su generosidad durante mi estadía será recordada con gratitud.

Además, no puedo dejar de mencionar a Nacho, quien me brindó una muy valiosa ayuda durante mi estancia en Duke. Su disposición para resolver dudas y colaboración con mi trabajo fueron de gran importancia para mí. Agradezco sinceramente su tiempo y esfuerzo.

A todas las demás personas que de alguna manera contribuyeron a este proyecto, ya sea brindando consejos, compartiendo recursos o simplemente brindando palabras de aliento, les estoy profundamente agradecido. Sus contribuciones fueron fundamentales para alcanzar los resultados obtenidos.

# RESUMEN

Esta tesis aborda el estudio de parte del *Archivo Berrutti*, un conjunto de documentos generados durante la última dictadura cívico-militar en Uruguay. El enfoque principal se centra en un grupo específico de estos documentos que consisten en fichas personales generadas por la O.C.O.A. (Organismo Coordinador de Operaciones Antisubversivas). El propósito fundamental de esta investigación es extraer la máxima cantidad de información posible de dichas fichas personales.

Para lograr este objetivo, se lleva a cabo un exhaustivo relevamiento del estado del arte en lo que respecta al análisis de documentos y el reconocimiento de texto. Posteriormente, se desarrolla una metodología basada en el empleo de técnicas de procesamiento de imágenes y aprendizaje automático, con el fin de extraer la información requerida de las fichas.

Es importante resaltar que esta tesis se enmarca en el proyecto CRUZAR, que persigue la creación y desarrollo de herramientas y metodologías para automatizar la extracción de información contenida en colecciones documentales sobre el pasado reciente en Uruguay.

Palabras clave:
Análisis de documentos históricos, Reconocimiento de patrones, OCR.

# ABSTRACT

This thesis addresses the study of part of the *Berrutti Archive*, a collection of documents generated during the last civic-military dictatorship in Uruguay. The focus is on a specific group of these documents, consisting of personal records generated by an organization called O.C.O.A. (Organismo Coordinador de Operaciones Antisubversivas). The fundamental purpose of this research is to extract the maximum amount of information possible from these personal records.

To achieve this objective, an exhaustive survey of the state of the art in document analysis and text recognition is conducted. Subsequently, a methodology is developed based on the use of image processing techniques and machine learning to extract the required information from the records.

It is essential to highlight that this thesis is part of the CRUZAR project, which aims to create and develop tools and methodologies to automate extracting information from documental collections about the recent history of Uruguay.



Keywords:
Historical document analysis, Template matching, OCR.

# Lista de figuras

# Lista de tablas

# Tabla de contenidos

# Capítulo 1

# Introducción

Entre los años 1973 y 1985, Uruguay fue gobernado por una dictadura cívico-militar que fue precedida por terrorismo de Estado entre los años 1968 a 1973. Durante estos períodos, ocurrieron violaciones a los derechos humanos, incluyendo secuestros, torturas, asesinatos y desapariciones. La investigación de estos crímenes ha sido difícil debido a complicidad de militares, políticos e instituciones involucradas en los hechos, que obstruyeron el acceso a las fuentes de información generadas durante ese período (Etcheverry et al. 2021).

Una de las fuentes documentales a la que se tiene acceso es el *Archivo Berrutti*. Esta es una colección de aproximadamente tres millones de imágenes escaneadas de microfilmaciones, agrupadas en rollos de aproximadamente 2000 imágenes cada uno (Etcheverry et al. 2021). Los documentos originales no se encuentran disponibles. Los rollos son bastante heterogéneos, con documentos generados por diversos organismos represores y organizaciones del Estado a lo largo de muchos años. La calidad de estos también es muy variada, existiendo imágenes de buena calidad e imágenes de mala calidad.

Los rollos del *Archivo Berrutti* están compuestos de hojas, cada una de ellas se corresponde con una imagen en la versión digital. Los rollos se encuentran numerados y el número de rollo es un dato que fue generado durante la microfilmación. Cada rollo cuenta con una imagen carátula que contiene el número de rollo. Existen

saltos en la numeración de los rollos y se desconoce el motivo.

Un conjunto de estos rollos contiene fichas personales (información personal y antecedentes) generadas por la O.C.O.A. (Organismo Coordinador de Operaciones Antisubversivas), uno de los principales organismos represores, creado con el fin de perseguir y capturar opositores («Organismo Coordinador de Operaciones Antisubversivas (OCOA) | Sitios de Memoria Uruguay», s.f.).

Debido al rol de este organismo, comprender el contenido de estos rollos es de suma importancia para comprender algunos de los hechos ocurridos durante la dictadura. Esta tarea se ve dificultada por el hecho de que los documentos se encuentran almacenados como imágenes lo que imposibilita el realizar búsquedas por texto. Además, la estructura y tipografía hacen que las herramientas de reconocimiento de texto no brinden buenos resultados al utilizar las imágenes en su estado actual como entrada.

Esta tesis se enmarca en CRUZAR («Cruzar – Archivos del pasado reciente», s.f.). CRUZAR es un proyecto que tiene como objetivo el uso de herramientas informáticas para ordenar, clasificar y comprender de manera sistemática el contenido de los diversos documentos generados durante la dictadura. El proyecto es llevado adelante por docentes, estudiantes y egresados de la Facultad de Información y Comunicación, la Facultad de Ciencias Sociales y la Facultad de Ingeniería de la UDELAR con apoyo de la organización Madres y Familiares de Detenidos Desaparecidos.

El objetivo principal de esta tesis es extraer la mayor cantidad de información posible de las fichas de O.C.O.A. En particular, extraer el contenido de los campos de las fichas y generar un sistema de búsqueda sobre las fichas de manera que la información pueda ser analizada fácilmente por historiadores, abogados, periodistas, familiares y otras personas interesadas. Dado que la información se encuentra en formato de imágenes, el proceso de búsqueda puede resultar arduo y demandar una considerable cantidad de tiempo.

Además del fichero general de la O.C.O.A. existen otros ficheros de organizaciones de inteligencia, como por ejemplo el fichero general del S.I.D. (Servicio de Información de Defensa). Estos ficheros comparten características con el de la

O.C.O.A. Se busca también que las técnicas aplicadas en el fichero de la O.C.O.A. puedan ser aprovechadas en estos otros ficheros en el futuro. Esto requiere que todas las herramientas desarrolladas estén correctamente documentadas y el código se encuentre lo más ordenado posible.

La mayoría de las fichas de individuos se componen de una parte frontal con información personal y una parte de atrás con anotaciones, antecedentes u observaciones. Existen casos en los que para un mismo individuo hay más de una parte de atrás o más de una parte frontal. Las fichas se generaron completando plantillas impresas, tanto las partes frontales como las de atrás.

En la Figura 1.1 hay una imagen de ejemplo de uno de los rollos del fichero general de la O.C.O.A. Se observa que en esta imagen hay dos partes frontales y dos partes de atrás. Para generalizar, se define 'parte ficha', 'parte de una ficha' o 'parte de ficha' a los rectángulos que contienen información relacionada a una ficha dentro de una hoja del rollo. Por ejemplo, la parte frontal de una ficha o parte de atrás de una ficha se consideran partes de una ficha. Todas las partes de ficha tienen la misma altura y el mismo ancho en todos los rollos. Además, cada hoja puede contener entre una y cinco partes de ficha. Junto con las partes frontales y de atrás, también puede haber partes de ficha con otra información como fotos o huellas dactilares.

La parte de atrás de una ficha puede estar ubicada en la hoja siguiente o en la misma hoja que la parte frontal. Como se muestra en el ejemplo de la Figura 1.1, se puede observar la parte de atrás de la ficha 307 (la parte frontal está ubicada en la hoja anterior), parte frontal de la ficha 308 y parte frontal y de atrás de la ficha 309. En ocasiones, es posible encontrar casos en los que la parte de atrás se encuentra vacía (completamente en blanco, incluso sin plantilla) o ausente.

En el listado 1.1 se encuentra el resultado de aplicar Tesseract (Smith, 2007), una herramienta open source para reconocer texto, a la imagen de la Figura 1.1. Se recortaron las últimas cuatro líneas del resultado para que ocupara una página sola. Lo que Tesseract detecta mejor son las etiquetas de los campos de la plantilla (Nombre, Ocupación, etc). En algunos casos se detecta también el contenido de los campos. Por ejemplo, se detecta correctamente el nombre 'Acosta de Rodriguez Zulma'. Sin embargo, es muy difícil saber qué texto pertenece a cada una de las

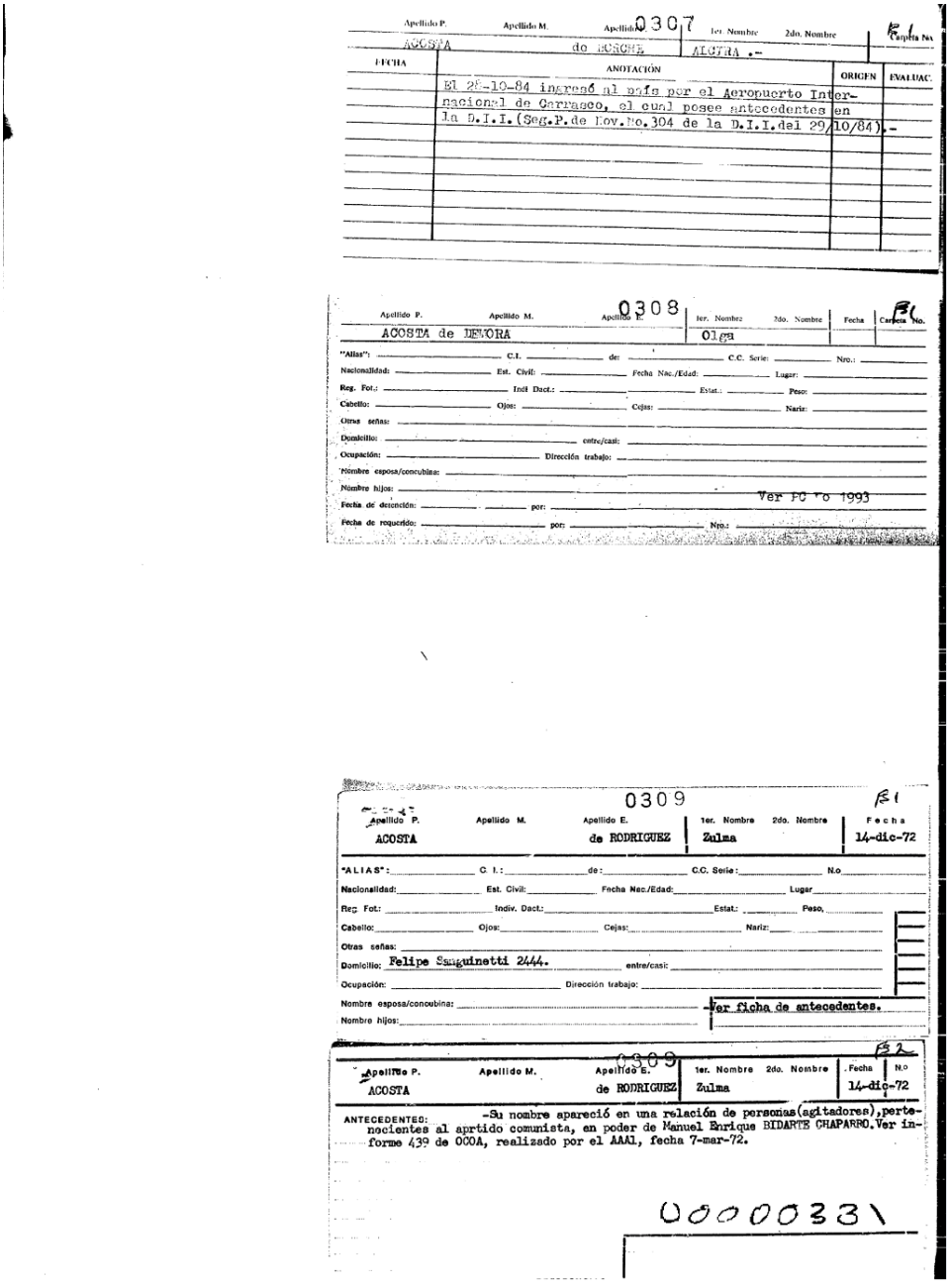


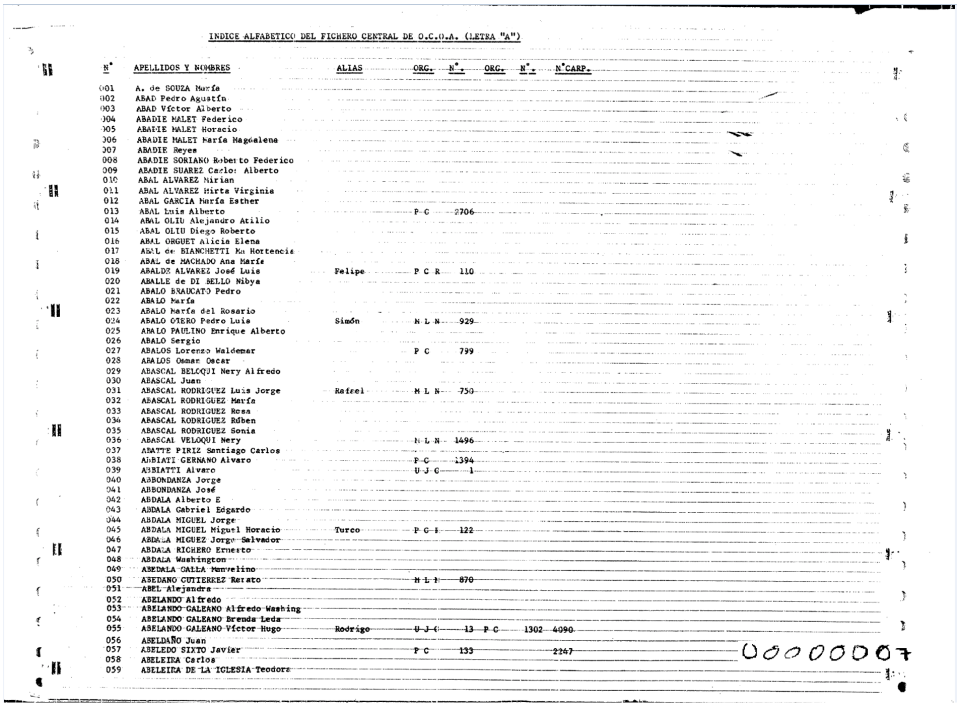

**Figura 1.1:** Ejemplo de una hoja con cuatro partes de ficha. Rollo 570 hoja 331.

partes de ficha así como también es complicado asociar el texto detectado a un campo.

```
A tr try
¡ALAN EIN
"Cabello:
. Ocupación:
uy
2d0. Nombre
ACOSTA de
LUNCH ALUTIRA .-
PEOHA
ANOTACIÓ
IÓN ORIGEN | EVALUAC.
al osufs por el Aer
21 de Carrasco
er--
en
10/873l=
T. dol
$4 cia
1
--- |
i_ _ _-__-_
A |
A
Apellido P. Apellido M. . 0] y 0 Jer. Nombra 2do0. Nombre | Fecha WI
(q (OAAMMMMMMAMAA24+4+< E ES e. e -- AAA
ACOSTA de DENFORA Olga | |
e¿qx- --_---<MIddMMMANAM05%55%5 O
> H .. Ñ : !
Alias". Ct de: C.C. Serie: -__ 2 N0.:
Nacionalidad: Est. Civil: Fecha Nac./Edad: Lugar
Indi Dact.: --.
Reg. Fot.: Estat: - Peso:
q¿_EE_eevv Entre casi: (AAA A
Dirección trabajo: - y
Nómbre esposa ocu: - _ _ __-_mmeooee A AA
Nombre hijos: lA A -
, 7 AAA UE Tag3
PS 1
- Fecha. dé detención:
- por:
Fecha de requerido: (O_OQOoxx por:
Do q
| _Apellido Pp. Apellido M. Apellido E. 1er. Nombre 2d0. tdombre | Fecha
ACOSTA de RODRIGUEZ Zulma
"ALIAS": Cc. |: de: a e. Señer 1 NI
Í Nacionalidad: Ñ _. Est. Civil: . Fecha Nac./Edad:____
os A IS A A IA
¡ Cabello: A A AN Nariz, a
i
a A A A
| Domicilio; Felipe Sauguinetti 2444.
 entre/casi:
t
: Ocupación:
_Nombre OIPOSBÍCONQUDN E. irc cc e Ver. ficha de antecedentes. A
: Nombre hijos: _.
" _APolIIMo P. UN Apellido M.. Ape aE | No
ACOSTA de RODRIGUEZ] Zulma 14-dig=72
mn -
ter. Nombre 2du0. Nombre . Fecha
$
El z y
l -Su nombre apareció en una relación de persoras(agitadores),perte-
Mocientes"al aprtido comunista, en poder de Manuel Enrique BIDARTE CHAPARBO, Ver in-!
--.- forme 439 de OCOA, realizado por el AAA, fecha 7-mar-72. l E
L0Io0O0OsS3N
DIECeEOOCraRIrÑriac.
ATA . i r a
Apellido P. Apellido M. api) al 0 É ler.
arptta Na
```

**Listing 1.1:** Resultado de aplicar Tesseract sobre la hoja 331 del rollo 570.

Acompañando a las fichas, existe un índice ordenado alfabéticamente, distribuido entre todos los rollos en formato tabla, que contiene información personal sobre los individuos. Otro objetivo es trabajar sobre este índice, para recortar las líneas y reconocer el texto ya que contiene información que no se encuentra en las fichas. En la Figura 1.2 se muestra una imagen de ejemplo de una hoja del índice.



INDICE ALFABETICO DEL FICHERO CENTRAL DE O.C.U.A. (LETRA "A")

| Nº | APELLIDOS Y NOMBRES | ALIAS | ORG. | Nº. | ORG. | Nº. | Nº CARP. |
|---|---|---|---|---|---|---|---|
| 001 | A. de SOUZA María | | | | | | |
| 002 | ABAD Pedro Agustín | | | | | | |
| 003 | ABAD Víctor Alberto | | | | | | |
| 004 | ABADIE MALET Federico | | | | | | |
| 005 | ABADIE MALET Horacio | | | | | | |
| 006 | ABADIE MALET María Magdalena | | | | | | |
| 007 | ABADIE Reyes | | | | | | |
| 008 | ABADIE SORIANO Robert to Federico | | | | | | |
| 009 | ABADIE SUAREZ Carlos Alberto | | | | | | |
| 010 | ABAL ALVAREZ Hirlan | | | | | | |
| 011 | ABAL ALVAREZ Hirte Virginia | | | | | | |
| 012 | ABAL GARCIA María Esther | | | | | | |
| 013 | ABAL Luis Alberto | | P C | 2706 | | | |
| 014 | ABAL OLIU Alejandro Atilio | | | | | | |
| 015 | ABAL OLIU Diego Roberto | | | | | | |
| 016 | ABAL ORGUET Alicia Elena | | | | | | |
| 017 | ABAL de BIANCHETTI Ma Hortencia | | | | | | |
| 018 | ABAL de MACHADO Ana María | | | | | | |
| 019 | ABALDO ALVAREZ José Luis | Felipe | P C R | 110 | | | |
| 020 | ABALLE de DI BELLO Nibya | | | | | | |
| 021 | ABALO BAUCAT Pedro | | | | | | |
| 022 | ABALO María | | | | | | |
| 023 | ABALO María del Rosario | | | | | | |
| 024 | ABALO OTERO Pedro Luis | Simón | N L N | 929 | | | |
| 025 | ABALO PAULINO Enrique Alberto | | | | | | |
| 026 | ABALO Sergio | | | | | | |
| 027 | ABALOS Lorenzo Waldemar | | P C | 799 | | | |
| 028 | ABALOS Omar Cesar | | | | | | |
| 029 | ABASCAL BELCQUI Mery Alfredo | | | | | | |
| 030 | ABASCAL Juan | | | | | | |
| 031 | ABASCAL RODRIGUEZ Luis Jorge | Rafael | N L N | 750 | | | |
| 032 | ABASCAL RODRIGUEZ María | | | | | | |
| 033 | ABASCAL RODRIGUEZ Rosa | | | | | | |
| 034 | ABASCAL RODRIGUEZ Ruben | | | | | | |
| 035 | ABASCAL RODRIGUEZ Sonia | | | | | | |
| 036 | ABASCAL VELOQUI Mery | | N L N | 1496 | | | |
| 037 | ABATTE PIRIZ Santiago Carlos | | | | | | |
| 038 | ABBIATI GERNANO Alvaro | | P C | 1394 | | | |
| 039 | ABBIATTI Alvaro | | U J C | 1 | | | |
| 040 | ABBONDANZA Jorge | | | | | | |
| 041 | ABBONDANZA José | | | | | | |
| 042 | ABDALA Alberto E | | | | | | |
| 043 | ABDALA Gabriel Edgardo | | | | | | |
| 044 | ABDALA MIGUEL Jorge | | | | | | |
| 045 | ABDALA MIGUEL Miguel Horacio | Turco | P C I | 122 | | | |
| 046 | ABDALA MIGUEZ Jorge Salvador | | | | | | |
| 047 | ABDALA RICHERO Ernesto | | | | | | |
| 048 | ABDALA Washington | | | | | | |
| 049 | ABDALA CALLA Manuelino | | | | | | |
| 050 | ABEDANO GUTIERREZ Renato | | N L I | 870 | | | |
| 051 | ABEL Alejandra | | | | | | |
| 052 | ABELANDO Alfredo | | | | | | |
| 053 | ABELANDO GALEANO Alfredo Washing | | | | | | |
| 054 | ABELANDO GALEANO Brenda Leda | | | | | | |
| 055 | ABELANDO GALEANO Víctor Hugo | Rodrigo | U J C | 13 | P C | 1302 | 4090 |
| 056 | ABELDAÑO Juan | | | | | | |
| 057 | ABELEDO SIXTO Javier | | P C | 133 | | 2247 | |
| 058 | ABELEIRA Carlos | | | | | | |
| 059 | ABELEIRA DE LA IGLESIA Teodore | | | | | | |

00000007

**Figura 1.2:** Ejemplo una hoja perteneciente al índice del rollo 570.

Una restricción adicional con la que se cuenta es que los documentos pertenecientes al *Archivo Berrutti* son reservados al momento de realizar esta tesis. Como consecuencia, no es posible utilizar herramientas o servicios externos que impliquen subir las imágenes a servidores de terceros, por lo que el procesamiento debe llevarse a cabo localmente o en servidores de confianza como Cluster.uy (Nesmachnow y Iturriaga, 2019).

Cluster.uy es una iniciativa nacional para proveer infraestructura con alto poder de cómputo para fomentar los proyectos de investigación e innovación. Cuenta con servidores con alto poder de cómputo, que incluyen equipos con tarjetas de vi-

deo. Físicamente, se encuentra ubicado en el datacenter 'Ing. José Luis Massera' de ANTEL.

Los resultados esperados de esta tesis son una base de datos que contenga la transcripción de los campos de las partes frontales de las fichas y de todas las líneas del índice. Junto a esto se espera desarrollar una interfaz visual que permita realizar consultas fácilmente a esta base de datos por nombre o cédula y que permita acceder a las imágenes originales fácilmente.

La principal contribución de esta tesis es el desarrollo de una metodología para la extracción y transcripción de los campos con información personal en las fichas generadas por la O.C.O.A. El enfoque utilizado se basa en técnicas clásicas de procesamiento de imágenes y herramientas de reconocimiento de texto. Además, se desarrolló un conjunto de herramientas que pueden ser utilizadas en otros ficheros del *Archivo Berrutti*.

Los siguientes capítulos se organizan de la siguiente manera: en el capítulo 2 se abordan los fundamentos teóricos que respaldan las técnicas aplicadas en este trabajo. A continuación, en el capítulo 3, se ofrece una descripción detallada de los datos con los que se trabajará. El capítulo 4 expone la metodología desarrollada para alcanzar los objetivos planteados. Se justifican las técnicas utilizadas y decisiones tomadas. Los resultados obtenidos de aplicar la metodología desarrollada se exponen y analizan en el capítulo 5. Por último, en los capítulos 6 y 7 se presentan las conclusiones de la tesis y se analizan posibles líneas de trabajo futuro.

# Capítulo 2

# Fundamentos teóricos

En este capítulo, se llevará a cabo una revisión de la literatura con el objetivo de proporcionar un marco teórico para la tesis. Se abordarán conceptos teóricos de relevancia y se presentarán estudios previos significativos, relacionados con la tesis realizada.

## 2.1. Reconocimiento de caracteres

El reconocimiento óptico de caracteres u OCR por su sigla en inglés consiste en un conjunto de diferentes algoritmos y técnicas de procesamiento de imágenes que tienen el objetivo de detectar y reconocer los caracteres presentes en una imagen y convertirlos en texto que se puede buscar, editar o almacenar electrónicamente. Los OCR tienen una amplia variedad de aplicaciones, como la digitalización de documentos impresos, reconocimiento de matrículas de autos, la lectura de facturas, recibos, y cheques entre otros (Doermann y Tombre, 2014).

En la actualidad, los OCR funcionan muy bien sobre texto moderno. No obstante, los datos utilizados para desarrollar los modelos en general no contienen documentos antiguos, algo necesario para obtener buen rendimiento en textos históricos. La calidad del material original, tipografía y la disposición de los elementos son

solo algunas de las dificultades que suelen presentar los textos antiguos de acuerdo con Nguyen et al. 2021.

Breuel et al. 2013 clasifica los sistemas de OCR en dos categorías según su método de funcionamiento. La primera categoría comprende los OCR basados en segmentación. Estos sistemas segmentan las líneas de texto en caracteres o candidatos a caracteres. Finalmente, se aplica un clasificador de caracteres a cada uno de los candidatos detectados. La otra categoría engloba a los OCR que no realizan segmentación por carácter y procesan el texto de a líneas.

De acuerdo con Breuel et al. 2013, el principal problema con los OCR basados en segmentación de caracteres es que requieren de un muy buen sistema de segmentación, ya que errores al segmentar los caracteres generalmente se transforman en errores de la salida. Los modelos sin segmentación no tienen este problema pero de todas formas tienen otros desafíos como tomar decisiones delicadas al desarrollar la estructura de los modelos.

Un ejemplo de OCR por segmentación es Tesseract (Smith, 2007) en sus versiones anteriores (en la actualidad, no utiliza segmentación). Ejemplos de modelos sin segmentación son los modelos ocultos de Markov (HMM) o modelos basados en redes neuronales como Calamari (Wick et al. 2020).

Según Doermann y Tombre, 2014, el funcionamiento de un OCR se puede dividir en tres etapas. La etapa de preprocesamiento incluye algunos pasos de preparación de la imagen. La etapa de extracción de características consiste en convertir cada línea de texto (o campo en el caso de esta tesis) y generar una secuencia de vectores de características. Finalmente, en la etapa de reconocimiento se aplican métodos de procesamiento de secuencias para reconocer el texto.

### 2.1.1. Preprocesamiento de imagen

Esta es una etapa de preparación de la imagen. Según Doermann y Tombre, 2014, un paso sugerido es binarizar la imagen. Esto significa convertir la imagen (a color o monocromática) a un solo canal con valores cero y uno. El objetivo de la

binarización es separar el texto del fondo. Existen diversos algoritmos desarrollados en el área. En el caso de los rollos de la O.C.O.A., las imágenes ya se encuentran binarizadas, por lo que no se entra en detalle en estos algoritmos. Es importante destacar que la binarización no es un paso necesario en todas las aproximaciones, pues implica pérdida de información en ciertos casos.

El próximo paso en la etapa de preprocesamiento consiste en detectar y enderezar documentos rotados. En el caso del fichero de la O.C.O.A., tanto las imágenes del índice como de las fichas pueden encontrarse rotadas. Algunos métodos de detección y corrección de rotación se describen más adelante en la sección 2.3.

Es este momento, el paso siguiente es la segmentación de página. Esto consiste en segmentar la página en regiones homogéneas de texto. Por ejemplo, tablas y columnas de texto. Doermann y Tombre, 2014, menciona que existen diversos algoritmos desarrollados con este objetivo. Se puede ver el problema de extraer todas las partes de ficha en una misma hoja como un problema de segmentación. En el caso del índice, hay un único bloque de texto por lo que este paso no es necesario.

Uno de los trabajos sobre segmentación de página es el de Pavlidis y Zhou, 1992. Los autores mencionan que entre dos bloques de texto, por ejemplo, dos columnas, existe un espacio en blanco que es mayor al espacio en blanco entre dos renglones. Proponen utilizar la proyección de perfil horizontal para detectar estos espacios. La proyección de perfil horizontal es un vector donde cada entrada contiene la suma de la cantidad de píxeles negros en esa fila de la imagen. Asimismo, existe la proyección vertical que es equivalente pero por columnas. Buscando espacios grandes en blanco (muchos valores en cero) en las proyecciones de perfil horizontal y vertical los autores logran segmentar hojas con texto en columnas e imágenes.

Finalmente, el último paso es segmentar las líneas (renglones) de texto y segmentar los caracteres en caso de que el OCR seleccionado lo requiera. Los campos de las fichas se extraen con la técnica descrita más adelante en la sección 2.2 y siempre tienen una sola línea por lo que no se realiza este paso. Sin embargo, sí es necesario para las imágenes del índice. Para el texto escrito por computadora (o máquina de escribir en el caso del índice) este problema en general se considera resuelto y existen varias formas de resolverlo (Doermann y Tombre, 2014). Una de ellas es el trabajo de Nagy et al. 1992. Esta técnica utiliza la proyección de perfil.

En el caso de tener dos líneas de texto, el espacio entre ellas se va a ver reflejado como un valor muy bajo en este vector. En caso de que no haya ruido en la imagen, el valor debería ser cero. Buscando estos valores mínimos, se puede detectar el espacio entre las líneas y separar la imagen.

### 2.1.2. Extracción de características

Existen diversos métodos para la extracción de características. De acuerdo con Doermann y Tombre, 2014, para que el sistema de OCR funcione correctamente, las características no deben ser muy sensibles al ruido y deben tener poder discriminativo (diferentes caracteres deben producir características diferentes). Seleccionar pocas características puede no ser suficiente para lograr este objetivo pero elegir muchas puede derivar en sistemas inestables o sobre ajustados.

Algunos de los métodos más relevantes en los últimos años son los basados en redes neuronales. A continuación, se realiza una introducción a estos métodos utilizando el libro de Goodfellow et al. 2016 como principal referencia.

Las redes neuronales prealimentadas o *feed forward* tienen como objetivo aproximar una función $f^*$. Por ejemplo, un clasificador $y = f^*(x)$ para cada entrada $x$ asigna una categoría $y$. Una red neuronal prealimentada define un mapeo $y = f^*(x;\theta)$ y aprende el valor de los parámetros (o pesos) $\theta$ que resultan en la mejor función de aproximación. Estos modelos se llaman *feed forward* debido a que la información fluye en un solo sentido.

Se llaman redes porque generalmente se representan como una composición de varias funciones. Por ejemplo, se puede tener $f^1$, $f^2$ y $f^3$ encadenadas para formar $f(x) = f^3(f^2(f^1(x))$. Se dice que $f^1$ es la primera capa y así sucesivamente. Generalmente, las diferentes funciones se componen de una operación lineal y de una operación no lineal, también llamada activación. Este tipo de redes recibe una entrada de largo fijo y su salida también es de largo fijo. En este tipo de red la operación lineal suele ser una multiplicación matricial y la suma de un valor. Por ejemplo, para la capa uno la operación lineal puede ser la siguiente: $f^1(x) = \theta_1 * x + b_1$. En este caso, los parámetros que se deben encontrar son $\theta_1$ también llamada

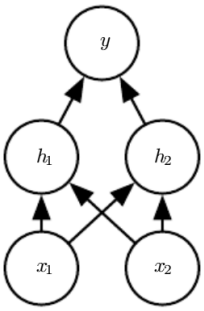



**Figura 2.1:** Ejemplo de una red neuronal de tres capas. Imagen tomada de Goodfellow et al. 2016.

matriz de pesos y el vector $b_1$ también conocido como vector de sesgos (la entrada $x$ es un vector). A las capas que realizan esta operación se les conoce como capas totalmente conectadas.

Existen varias funciones de activación. Una de las más utilizadas es la función ReLU (*rectified linear unit*), definida como $relu(x) = \max(0, x)$.

En la Figura 2.1 se encuentra la representación en grafo de una red neuronal de tres capas. La entrada tiene dos componentes o neuronas, la segunda capa también tiene dos neuronas y finalmente hay una única neurona de salida.

Para obtener los mejores parámetros (también conocido como aprender los parámetros), el algoritmo más utilizado es el de propagación hacia atrás o *backpropagation*. Este algoritmo resuelve un problema de optimización. Se debe contar con un conjunto de datos (llamados datos de entrenamiento), para los que se conoce su salida esperada (por ejemplo, para clasificación conocer su categoría). Además, se debe contar con una función de costo derivable cuyo mínimo debe darse cuando la salida de la red es la esperada en todos los casos. El algoritmo de propagación hacia atrás consiste en calcular la salida de la red para el conjunto de entrenamiento y minimizar la función de costo mediante el uso de descenso por gradiente, ajustando en cada capa los parámetros.

Las redes neuronales convolucionales son un tipo de red neuronal que funciona muy bien al trabajar con datos que se representan en formato matricial. Por ejemplo, una imagen, que se representa como una matriz de dos dimensiones. Estas redes

tienen por lo menos una capa en la que la operación lineal que realizan es una convolución.

Considerando una imagen $I$, la convolución $S$ (ambos de dos dimensiones) entre esa imagen y un núcleo $K$ se define de la siguiente manera:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{2.1}$$

En estas redes, mediante el algoritmo de *backpropagation* aprenden los valores de los diferentes núcleos $K$. Cada una de las capas de convolución contiene varios núcleos. El tamaño y cantidad de los núcleos a utilizar es algo a definir en el momento de elegir la arquitectura de la red.

En muchas implementaciones, en lugar de utilizar la convolución se utiliza la correlación cruzada, que es igual a la convolución pero sin invertir el núcleo. Esto resulta más fácil de implementar y no afecta el resultado, ya que el algoritmo de aprendizaje va a aprender los núcleos invertidos. En la siguiente fórmula se encuentra la ecuación utilizando correlación cruzada en lugar de convolución.

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(m,n)K(i+m, j+n) \tag{2.2}$$

Las ecuaciones 2.1 y 2.2 fueron tomadas del libro de Goodfellow et al. 2016.

En la Figura 2.2 se observa un ejemplo del cálculo de la convolución (sin invertir el núcleo) de una entrada con un núcleo. Como puede observarse, el núcleo se mueve por toda la entrada, multiplicando el núcleo por los elementos de la entrada y sumándolos. La cantidad de posiciones que se mueve el núcleo en cada etapa se llama *stride* y en este caso es 1.

Junto con la convolución, estas redes neuronales también realizan otra operación llamada *pooling*. Esta operación tiene como objetivo reducir la dimensión y funcionan de manera similar a las convoluciones solo que en lugar de devolver la

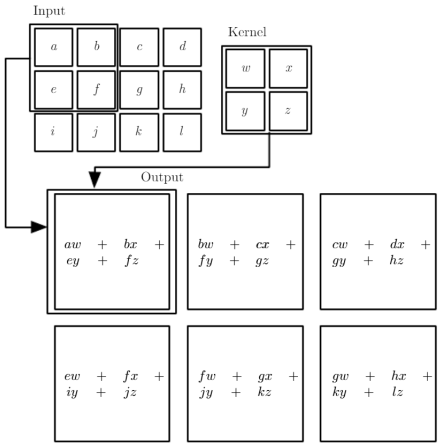



**Figura 2.2:** Ejemplo de una convolución de dos dimensiones. Imagen tomada de Goodfellow et al. 2016.

suma del producto de la imagen con el núcleo, devuelven el máximo en un área o el promedio (max y *average pooling*).

Estas redes demostraron ser muy buenas en tareas de clasificación de imágenes, ganando la competencia ImageNet con un error de 15 % el año que se utilizaron por primera vez, obteniendo el segundo lugar un error de 26 % (Krizhevsky et al. 2012).

Las redes neuronales convolucionales pueden utilizarse como extractor de características de un OCR. El trabajo de Rawls et al. 2017 presenta un sistema de OCR que utiliza redes neuronales convoluciones como extractor de características. Los autores utilizan una red con varias capas de convolución y de *max pooling*. La entrada de la red es una imagen binaria (dos dimensiones) y la salida de la última capa convolucional es de tres dimensiones, cuyo tamaño depende la cantidad de filtros en la última capa y la cantidad de *pooling* en toda la red. Esta salida de tres dimensiones se conecta a una capa totalmente conectada generando una secuencia de vectores de características de menor dimensión que se utilizan como entrada del método de reconocimiento.

### 2.1.3. Métodos de reconocimiento

Una vez obtenida la secuencia de vectores de características extraídos de la imagen, es necesario transformar esa secuencia de vectores en texto. Los modelos de Markov ocultos fueron utilizados durante mucho tiempo como métodos de reconocimiento. Se puede mencionar como ejemplo el trabajo desarrollado por Arica y Yarman-Vural, 2002. Estos métodos de reconocimiento fueron desplazados en la actualidad por redes neuronales. Nuevamente el libro de Goodfellow et al. 2016 se utiliza como principal referencia en esta sección.

El reconocimiento utilizando una red neuronal consiste en tomar los vectores de características generados de la imagen (que pueden haber sido generados con una red neuronal o con otro método), utilizarlos como entrada en una red neuronal y luego interpretar la salida de la red como palabras o caracteres.

Utilizando redes neuronales prealimentadas solamente una ventana de tamaño fijo de la secuencia de características puede utilizarse en cada posición. La información contextual, algo muy importante en esta tarea, solo puede utilizarse limitadamente. Las redes neuronales recurrentes son un tipo de red neuronal especializadas en procesar una secuencia de valores $x^{(1)}$, $x^{(2)}$,...,$x^{(\tau)}$ de largo variable. La idea detrás de este tipo de red neuronal es el compartir parámetros. Si se tuvieran diferentes parámetros para cada índice de tiempo, entonces la red no sería capaz de generalizar a secuencias de largo no vistas durante el entrenamiento (Goodfellow et al. 2016).

Estas redes mantienen un estado o memoria, llamado $h$ que es lo que permite relacionar el estado de tiempo $t$ con los anteriores. En general, se comienza con un estado inicial $h^{(0)}$ y para cada paso de tiempo desde $t = 1$ hasta $t = \tau$ se aplican las siguientes ecuaciones de actualización:

$$\begin{aligned} a^t &= b + Wh^{t-1} + Ux^t \\ h^t &= \tanh a^t \\ o^t &= c + Vh^t \\ y^t &= softmax(o^t) \end{aligned} \tag{2.3}$$

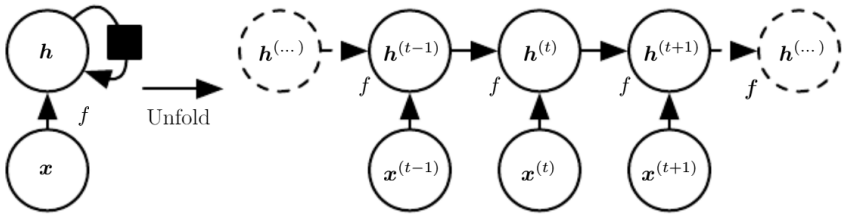



**Figura 2.3:** Ejemplo de una red neuronal recurrente, que procesa una entrada $x^{(t)}$, al procesar la entrada, el estado oculto $h$ se va actualizando. En la izquierda se observa un diagrama tipo circuito. En la derecha, se observa la misma red procesando una secuencia pero con el grafo computacional expandido, donde cada nodo está asociado con un instante de tiempo particular. Imagen tomada de Goodfellow et al. 2016.

Los parámetros que se aprenden son las matrices de pesos $U$, $V$ y $W$ y los vectores $b$ y $c$ que son los sesgos. Las funciones tanh (tangente hiperbólica) y softmax son las activaciones. Este es un ejemplo de una red neuronal recurrente que devuelve una salida del mismo tamaño que la entrada. La salida de la red en cada paso es $y^t$. El entrenamiento de estas redes se hace con una versión adaptada del algoritmo de propagación hacia atrás llamado propagación hacia atrás en el tiempo que tiene en cuenta el costo en cada paso del tiempo.

La función $softmax$ para vector $x$ de tamaño $K$ para la componente $j$ se encuentra definida en la Ecuación 2.4.

$$softmax(x)_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}} \tag{2.4}$$

En la Figura 2.3 se encuentra una red neuronal recurrente de ejemplo. A la izquierda se observa su representación en forma de diagrama, la neurona recibe una entrada $x^t$, la procesa y actualiza el estado oculto $h$. En la derecha, se observa el grafo computacional expandido, cada instante de tiempo se representa por nodos diferentes, mostrando el cambio del estado oculto $h$ y la entrada $x^{(t)}$.

No obstante, este tipo de redes neuronales sufren de un problema llamado desvanecimiento de gradientes. Este problema consiste en que los gradientes de la red neuronal crecen demasiado o se vuelven muy pequeños conforme van aumentando

las etapas de tiempo, lo que dificulta el entrenamiento. Como consecuencia, estas redes solo logran almacenar información por unas pocas etapas del proceso.

Las redes neuronales recurrentes LSTM (*long short-term memory*) no sufren este problema. Las arquitecturas LSTM se componen de un conjunto de subredes recurrentes conocidas como bloques de memoria. Cada bloque contiene una o más celdas de memoria y tres compuertas multiplicativas. Las compuertas de entrada, salida y de olvido. Estas compuertas multiplicativas permiten a la celda LSTM mantener y acceder a información por largos períodos de tiempo mitigando el problema de los gradientes desvanecientes. Por ejemplo, si la compuerta de entrada se mantiene cerrada (es decir, tiene una activación cercana a 0), no van a agregarse datos nuevos a la celda y por lo tanto la información existente puede utilizarse mucho más adelante en el tiempo abriendo la compuerta de salida. La compuerta de olvido decide qué información previa descartar (Graves, 2012).

En el modelado de secuencias, en ocasiones la información se encuentra distribuida a lo largo de múltiples pasos. Las redes neuronales recurrentes permiten utilizar información de pasos anteriores, pero resulta muy útil también utilizar información del futuro. Una red neuronal LSTM bidireccional sirve para cumplir este propósito. Estas redes constan de dos ramas que procesan la secuencia por separado. Una lo hace de izquierda a derecha y la otra de derecha a izquierda. La salida de ambas redes se combina y se obtienen los caracteres de salida.

En el trabajo publicado por Breuel et al. 2013, se genera un modelo OCR utilizando redes neuronales LSTM bidireccionales. Los conjuntos de datos que utilizan contienen texto impreso escrito por computadora. Los autores obtienen buenos resultados cuando se comparan con los otros OCR existentes en la época y destacan la facilidad de entrenamiento de estas redes.

En el ya mencionado trabajo de Rawls et al. 2017, luego de extraer las características usando redes neuronales convoluciones se utilizan redes neuronales LSTM bidireccionales para procesar la secuencia de vectores de características. Los autores mencionan que alcanzaron resultados del estado del arte utilizando solamente redes neuronales en el sistema, algo que mencionan como una novedad. Además, trabajan con varios conjuntos de datos que contienen tanto texto escrito a mano como escrito por máquina. La extracción de características y reconocimiento se realizan utilizan-

do la misma red neuronal, conectando la salida de la última capa convolucional con la entrada de la red LSTM. El entrenamiento se realiza en conjunto. A la salida de la última capa de la red LSTM hay una capa con tantas neuronas como el tamaño del alfabeto. Esta salida utiliza una función *softmax* como activación lo que genera una distribución de probabilidad de caracteres.

La función de costo que se utiliza es CTC *(Connectionist Temporal Classification)* (Graves et al. 2006). Esta función de costo ha demostrado muy buenos resultados en problemas de reconocimiento de secuencias y en particular en OCR. Soluciona un problema que tenían las redes neuronales recurrentes de tener que alinear la secuencia de entrada con la secuencia de salida, algo muy difícil en la práctica ya que implica conocer para cada vector de la secuencia de entrada a la red a que carácter corresponde en la imagen. Gracias a esta función de costo, esto ya no es necesario.

### 2.1.4. Herramientas seleccionadas

Para el desarrollo de esta tesis se seleccionaron dos OCR para reconocer texto. La primera herramienta seleccionada es Calamari (Wick et al. 2020), una herramienta open source que utiliza redes neuronales para reconocer texto, tomando como entrada líneas de texto. Fue desarrollada en Python y utiliza la biblioteca de aprendizaje profundo TensorFlow («TensorFlow», s.f.). Utiliza redes neuronales convolucionales como extractor de características y redes neuronales LSTM como método de reconocimiento. La biblioteca permite cambiar fácilmente la arquitectura al entrenar un modelo nuevo. En este trabajo se utiliza la arquitectura por defecto.

Los autores también exploran la técnica de *finetuning*. También conocida como o ajuste fino. Esta técnica surge como una mitigación del problema de las redes neuronales de precisar una gran cantidad de datos para su entrenamiento. Consiste en entrenar una red neuronal para una tarea con un conjunto de datos y luego tomar los parámetros resultantes de la red y utilizarlos como parámetros iniciales para entrenar la red con otros datos y para otra tarea, en ocasiones modificando los pesos de solo algunas capas o agregando capas nuevas al final. En este caso, toman modelos entrenados con una gran cantidad de líneas de texto (un conjunto de datos con

70.000 líneas y uno con 192.000) y luego entrenan en otros conjuntos con menos líneas de texto (tres conjuntos con entre 1.500 y 4.000). Realizan una comparación entre los modelos entrenados desde cero y cuando hicieron ajuste fino obteniendo mejores resultados al realizar ajuste fino. El tiempo de entrenamiento se encuentra optimizado debido a que soporta entrenamiento utilizando GPU. Se utiliza la versión 2.2.2, que es la última disponible. Este OCR ya fue utilizado en anteriores investigaciones sobre el *Archivo Berrutti* obteniendo buenos resultados (Etcheverry et al. 2021).

La otra herramienta que se utiliza es Tesseract (Smith, 2007), un OCR que es en la actualidad open source. Fue creado en 1984 por HP, volviéndose open source en 2005. Entre 2006 y 2018 fue desarrollado por Google. Su última versión mayor es la 5 («Tesseract User Manual», s.f.). Fue lanzada en noviembre de 2021 y es la que se utiliza en este trabajo. Utiliza redes neuronales de tipo LSTM, similar a Calamari pero no soporta entrenamiento o inferencia utilizando GPU, lo que lo hace más lento. En el caso de Tesseract, se utiliza la versión 5.3.0. No se encontró información detallada acerca de la arquitectura utilizada por Tesseract en esta versión en la documentación, se utiliza la arqutiectura por defecto.

Tesseract trae varios modos de ejecución, cada uno de ellos realiza diferentes tareas de preprocesamiento antes de reconocer el texto. Por ejemplo, hay modos que detectan la orientación del texto o detectan renglones. Asimismo, existe un modo para detectar texto vertical. Durante esta tesis, debido a que en todos los casos la imagen utilizada tiene una línea sola de texto, el modo utilizado (*page segmentation mode* o PSM) es el 13. Este modo trata la imagen como una línea de texto y no realiza ningún preprocesamiento.

Ambos OCR cuentan con modelos entrenados en grandes volúmenes de texto. No obstante, en general estos modelos no funcionaron bien al probarlos en los rollos de la O.C.O.A. (como se pudó ver en el Capítulo 1) por lo que se optó por etiquetar datos y entrenar modelos propios.

## 2.2. Reconocimiento de patrones

Esta sección se encuentra basada en su totalidad en el capítulo 23 del libro Burger y Burge, 2016 que trata este tema.

El reconocimiento de patrones también conocido como *template matching* es un problema que consiste en dada una imagen de búsqueda $I$ y una imagen de referencia (o imagen patrón) $R$, encontrar la posición de $I$ donde su contenido es exactamente igual o muy parecido a $R$. Esta técnica es utilizada en esta tesis tanto para clasificar las partes de ficha como para detectar y extraer los campos de texto. Se seleccionó debido a su facilidad de implementación y se obtuvieron buenos resultados.

Si se define

$$R_{r,s}(u, v) = R(u - r, v - s) \tag{2.5}$$

a la imagen de referencia desplazada por la distancia (r, s) en los ejes horizontales y verticales respectivamente, entonces el problema puede resumirse como:

Dada una imagen de búsqueda $I$ y una imagen de referencia $R$, encontrar los índices $(r, s)$ tales que la similitud entre la imagen de referencia desplazada $R_{r,s}$ y la correspondiente subimagen de $I$ sea máxima.

Esto requiere definir una métrica de similitud o distancia $d(R_{(r,s)}, I)$ entre la imagen de referencia $R$ desplazada $(r, i)$ y la imagen $I$. Por ejemplo, se puede utilizar distancia coseno, la norma L1 o la norma euclidiana. Por ejemplo, esta última se encuentra en la Ecuación 2.6.

$$d(R_{(r,s)}, I) = \left[ \sum_{(i,j)\in R} (I(r + i, s + j) - R(i, j))^2 \right]^{1/2} \tag{2.6}$$

La distancia euclidiana de N dimensiones de la Ecuación 2.6 tiene muy buenas propiedades. Para encontrar la mejor posición, alcanza con minimizar el cuadrado de $d(R_{(r,s)}, I)$ ya que la ecuación es siempre positiva.

El resultado de expandir el cuadrado se encuentra en la ecuación 2.7. El segundo término es la suma del cuadrado de los píxeles en la imagen de referencia. Este término es constante y por lo tanto no afecta a la hora de minimizar.

$$d^2(R_{(r,s)}, I) = \sum_{(i,j)\in R} I^2(r+i, s+j) + \sum_{(i,j)\in R} R^2(r+i, s+j) - 2\cdot \sum_{(i,j)\in R} I(r+i, s+j)\cdot R(i,j) \tag{2.7}$$

El primer término es la suma del cuadrado de la subimagen de $I$ generada con el desplazamiento $(r, s)$. El último término es la correlación cruzada ($\star$) entre $I$ y $R$, definida en el caso general en la Ecuación 2.8.

$$(I \star R)(r,s) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(r+i, s+j) \cdot R(i,j) \tag{2.8}$$

Asumiendo que $R$ e $I$ tienen $0$ (cero) en los valores afuera de los límites, entonces la Ecuación 2.8 es equivalente a la Ecuación 2.9. Esto es el último término de la Ecuación 2.7 (sin tener en cuenta el $2$ multiplicando). Este valor puede calcularse de manera eficiente en el espacio de frecuencia.

$$(I \star R)(r,s) = \sum_{(i,j)\in R} I(r+i, s+j) \cdot R(i,j) \tag{2.9}$$

Si se asume que el primer término de 2.7, que es la intensidad de $I$ en la subimagen definida por los índices $(r, s)$, es constante, entonces el lugar donde se da la máxima correlación cruzada coincide con el lugar donde se da la mayor similitud entre el patrón y la imagen de referencia. Desafortunadamente, en general el primer término no es constante y la correlación cruzada cambia mucho con cambios en la

intensidad de la imagen.

Una forma de tener en cuenta las variaciones en la intensidad de la imagen es definiendo la correlación cruzada normalizada en la Ecuación 2.10.

$$C_N(r,s) = \frac{\sum_{(i,j)\in R} I(r+i,s+j)\cdot R(i,j)}{\left[\sum_{(i,j)\in R} I^2(r+i,s+j)\right]^{1/2}\cdot\left[\sum_{(i,j)\in R} R^2(i,j)\right]^{1/2}} \tag{2.10}$$

Si las imágenes de referencia y el patrón son siempre positivas, entonces la correlación cruzada normalizada se encuentra en el rango de 0 a 1, donde 1 representa la máxima similitud entre $R$ y la subimagen de $I$ generada por los índices $(r,s)$ mientras que si es 0 entonces la similitud es mínima.

Sin embargo, esta nueva correlación normalizada tiene el problema de que compara de manera absoluta la distancia entre el patrón y la subimagen. Si la intensidad de la imagen se altera, entonces la correlación cruzada normalizada puede cambiar demasiado. Una solución a este problema es utilizar en lugar de $R$ y la subimagen de $I$, las diferencias con respecto al promedio de $R$ y la subimagen de $I$. Estos cambios se encuentran en la Ecuación 2.11.

$$C_L(r,s) = \frac{\sum_{(i,j)\in R}\left(I(r+i,s+j)-\bar{I}_{r,s}\right)\cdot\left(R(i,j)-\bar{R}\right)}{\left[\sum_{(i,j)\in R}\left(I(r+i,s+j)-\bar{I}_{r,s}\right)^2\right]^{1/2}\cdot\left[\sum_{(i,j)\in R}\left(R(i,j)-\bar{R}\right)^2\right]^{1/2}} \tag{2.11}$$

$$\bar{R} = \frac{1}{K}\cdot\sum_{(i,j)\in R} R(i,j) \tag{2.12}$$

$$\bar{I}_{r,s} = \frac{1}{K}\cdot\sum_{(i,j)\in R} I(r+i,s+j) \tag{2.13}$$

Siendo $K$ la cantidad de píxeles de $R$. A esta nueva ecuación se le conoce como coeficiente de correlación. Los valores se encuentran entre $-1$ y $1$ donde $1$ representa similitud máxima y $-1$ representa que el patrón y la subimagen son lo más diferentes posibles.

De acuerdo con Burger y Burge, 2016 estos métodos de reconocimiento de patrones basados en correlación no son capaces de manejar de manera correcta rotaciones o escalados del patrón en la imagen. Una solución que puede resultar poco eficiente consiste en probar con el patrón rotado en diferentes ángulos y a diferentes escalas. Esta es la aproximación que se utiliza en esta tesis ya que los datos tienen todos la misma escala, la rotación no es muy grande y además esta técnica es sencilla de implementar.

## 2.3. Detección de documentos rotados

Según Papandreou et al. 2013, al momento de aplicar OCR a una imagen, un paso de preprocesamiento muy importante es detectar si el documento se encuentra rotado. La performance de un OCR puede verse reducida si el texto en la imagen de entrada se encuentra rotado.

El trabajo realizado por Al-Khatatneh et al. 2015 realiza una comparación de tres métodos populares para detección de imágenes rotadas. El primer método se conoce como proyección de perfil, el segundo como transformada de Hough y por último están los métodos de vecinos cercanos. A continuación, se realiza una breve descripción de cada método y las conclusiones a las que arriba el artículo mencionado. En los tres métodos siempre el primer paso es binarizar la imagen.

### 2.3.1. Proyección de perfil

Esta técnica de detección de documentos rotados fue descrita inicialmente por Postl, 1986. Utiliza la proyección de perfil horizontal definida previamente en la sección 2.1.1.

Para documentos derechos, es decir, rotados con ángulo 0, la proyección horizontal tiene valles que se corresponden con los espacios entre las líneas y los picos se corresponde con las líneas de texto en la imagen. El método que propuso Postl, 1986 consiste en calcular la variación de la proyección horizontal rotando la imagen por varios ángulos. El ángulo con la variación más alta es el ángulo por el cual el documento se encuentra rotado. Una vez obtenido este ángulo, se puede rotar la imagen por ese ángulo en el sentido contrario para enderezar la imagen.

En Al-Khatatneh et al. 2015 se menciona que este método puede resultar muy costoso computacionalmente. Algunos autores desarrollaron variaciones de este método con el objetivo de mejorar su eficiencia. También se menciona que el rendimiento puede bajar si el documento contiene imágenes o diagramas y es un método muy sensible al ruido.

Los autores concluyen que este método fue el que les dio la mejor estimación del ángulo de rotación pero fue el más lento en tiempo de ejecución de los tres. En vista de estos resultados y su facilidad de implementación, para la tesis se opta por utilizar este método.

### 2.3.2. Transformada de Hough (HT)

La transformada de Hough es una técnica de extracción de características muy utilizada en el análisis de imagen. Sirve para extraer características de cierta forma, como curvas o líneas especificadas de forma paramétrica.

Fue introducida como una transformada lineal para detectar líneas rectas en imágenes. En una imagen, cualquier recta puede escribirse como $y = mx + b$ donde $m$ representa la pendiente y $b$ la distancia al origen.

La idea detrás de la transformada de Hough es representar líneas rectas en términos de sus parámetros de pendiente y distancia al origen en lugar de utilizando puntos. La recta $y = mx + b$ se representa como $(m, b)$ en el espacio de parámetros. Otra forma de parametrizar una recta es utilizando coordenadas polares $\rho$ y $\theta$. $\rho$ representa la distancia entre la recta y el origen y $\theta$ representa el ángulo entre los

vectores del origen y el punto más cercano de la recta.

Cada punto $(x, y)$ de la imagen en el espacio cartesiano se mapea a coordenadas polares en el espacio de Hough utilizando la siguiente función: $\rho = x \cos\theta + y \sin\theta$.

Es decir, cada punto de la imagen se transforma en todas las posibles rectas que pasan por él. Luego, se buscan los valores de $(\rho, \theta)$ donde más curvas coincidan. Esto significa que esos puntos juntos forman una línea recta. Finalmente, el ángulo de rotación de la imagen se puede calcular como el $\theta$ promedio de las líneas detectadas con más puntos.

Según Al-Khatatneh et al. 2015, este método tiene un buen desempeño en general pero la ocurrencia de imágenes, títulos, columnas o pies de página dificultan la tarea de encontrar las líneas rectas. Además, el ruido aumenta el tiempo de ejecución y es un método que requiere de mucha memoria para su ejecución.

### 2.3.3. Métodos de vecinos cercanos

Según Lu y Tan, 2003, el primer método de detección de texto rotado de vecinos cercanos fue propuesto por Hashizume et al. 1986. Las imágenes contienen pequeñas componentes que se encuentran alineadas en cierta dirección como pueden ser los caracteres en un renglón de texto. Este método consiste primero en encontrar esas componentes. Luego, para cada componente se busca su componente vecina más cercana. A continuación, para cada par de vecinos más cercanos se calcula el vector de dirección y el ángulo de ese vector se almacena en un histograma. Una vez finalizado el cálculo para todos los pares de componentes, el pico en el histograma indica el ángulo dominante de rotación de la imagen.

Al-Khatatneh et al. 2015 menciona que a lo largo de los años se desarrollaron muchas variantes de esta técnica, con bajos tiempos de ejecución y buen desempeño. También se menciona que estos métodos son muy dependientes del proceso de binarización utilizado y por lo tanto tiene problemas en documentos históricos o de baja calidad. De los métodos que compararon, este resultó ser el más rápido. Mencionan que funciona bien con documentos rotados en un rango bastante amplio

de ángulos.

## 2.4. Trabajos relacionados

En esta sección se analizan algunos trabajos relacionados con extracción de información de documentos.

En el trabajo publicado por Gorski et al. 2001, se desarrolla un sistema para reconocer montos de cheques en inglés o francés. Desarrollan un sistema para documentos manuscritos y uno por separado para documentos impresos. Ambos sistemas son capaces de detectar los diferentes campos en los cheques y luego se aplican técnicas de segmentación para obtener los números o caracteres. Finalmente, aplica OCR sobre el resultado. El ratio de reconocimiento en producción varía de un 65 % a un 85 %. Se menciona que el sistema es muy sensible a pequeños cambios en la tipografía de diferentes idiomas, incluso cuando los caracteres son los mismos, en dos países pueden escribirse de manera diferente causando errores. Un ejemplo mencionado en el artículo es que el número 4 escrito en Francia es parecido al número 6 en Estados Unidos.

En la investigación por Martínek et al. 2020 se desarrolla un sistema para aplicar OCR a documentos históricos impresos, generalmente de diarios, en alemán. El sistema primero realiza una segmentación para detectar bloques de texto, luego realiza una segmentación de renglones y finalmente aplica OCR a cada línea detectada. Para la segmentación tanto de bloques como de líneas se utilizan redes neuronales convolucionales. Para el OCR utilizan un extractor de características con redes convolucionales y utilizan redes neuronales LSTM para el reconocimiento. Utilizando pocos datos etiquetados los autores alcanzan muy buenos resultados en reconocimiento de texto. Mencionan que se alcanzan resultados al mismo nivel que el estado del arte, logrando desarrollar un OCR muy eficiente específico para textos históricos en alemán.

Un trabajo reciente por Dhakal et al. 2019 utiliza una combinación de template matching y similitud de texto para clasificar documentos (varios tipos de facturas) y extraer campos relevantes de ellos. Se genera una base de datos de plantillas de

imágenes y texto (el texto es común a todos los documentos de la misma plantilla). Al momento de clasificar una imagen candidata nueva, se calcula la similitud visual y textual para todas las plantillas. La plantilla que obtenga la similitud visual y textual sumada más alta y que además supere cierto umbral es seleccionada como la plantilla del documento nuevo. En caso de no haber coincidencia, se envía la imagen para un etiquetado manual. Los autores seleccionaron «Extracción inteligente de texto y datos con OCR - Amazon Textract - Amazon Web Services», s.f. de Amazon como OCR.

Para la similitud visual, los autores calculan la distancia coseno entre las matrices $\sigma$ de la descomposición SVD de la imagen nueva y las imágenes plantilla. Para la similitud de texto, se aplica un OCR a la imagen candidata y luego se calcula el promedio de la distancia de edición entre las primeras y últimas $n$ líneas del texto obtenido de la imagen con el texto de la plantilla.

Una vez se determina la plantilla correcta para el nuevo documento, se seleccionan las regiones de interés donde están los campos a extraer. Este proceso se realiza por separado para cada campo. Primero, se recorta el campo en la imagen plantilla (requiere como paso previo anotar manualmente en las plantillas la posición de cada campo). Luego, se busca la zona de la imagen candidata que tenga la mayor similitud al campo recortado de la plantilla utilizando la correlación. En este punto, se realiza una estimación de la ubicación del campo a extraer utilizando el texto en común en la plantilla y la imagen candidata. Finalmente, se toma el texto del OCR cuya posición se encuentre dentro de donde se estimó que se encontraba el campo.

En promedio, los autores obtienen un 86 % de accuracy. Consideran que esto es un muy buen resultado ya que no se requiere generar reglas para clase (si no que alcanza con un documento plantilla), utiliza pocos ejemplos de entrenamiento y, además, generaron una herramienta que permite revisar los resultados y corregir fácilmente errores. Por lo tanto, este sistema puede reducir los costos de empresas de una manera significativa. También destacan que solo se necesita un ejemplo de documento de cada clase para poder generar la plantilla. Una posible mejora que mencionan es utilizar varias plantillas para una misma clase de documento con la idea de que esto puede mejorar el rendimiento.

# Capítulo 3

# Descripción de los datos

En el Capítulo 1 se menciona que los datos a trabajar son el fichero general de la O.C.O.A. perteneciente al *Archivo Berrutti*. Asimismo, se menciona que las fichas de las personas se encuentran agrupadas en rollos. En este capítulo, se analiza más en detalle el contenido de estos rollos, se muestra cómo están distribuidos los datos y se analizan posibles problemas de calidad de datos que presentan las imágenes.

El fichero general de la O.C.O.A. consta de varios rollos y se encuentra ordenado alfabéticamente por apellido. Para cada letra del alfabeto, existe un único rollo que contiene todas las fichas de personas cuyo primer apellido comienza con esa letra. Además, algunos rollos contienen más de una letra. Como fue mencionado en el capítulo 1, junto con las fichas cada rollo contiene un índice (para cada letra en él) en formato tabla que contiene un listado de personas ordenado alfabéticamente acompañado de otra información relevante acerca de la persona: el número de ficha, organizaciones a las que pertenecía la persona (hasta un máximo de dos organizaciones), su alias y observaciones. Finalmente, el contenido de cada rollo se completa con una hoja carátula al comienzo y algunas hojas con metadatos sobre la microfilmación, que contienen, por ejemplo, la fecha de realizada.

Existen algunos rollos digitalizados varias veces (algunos hasta tres veces). Esto genera que se tengan datos duplicados pero con diferente calidad. En las situaciones en las que se cuenta con más de un rollo, para trabajar con las fichas se realiza una

selección manual del rollo que presenta una mejor calidad de digitalización. En general, la diferencia de calidad es muy notoria. En la Figura 3.1 se muestran las dos versiones de una misma hoja del rollo 588, el cual cuenta con dos versiones. Se observa que la versión de la izquierda se lee mucho mejor. Al procesar el índice, se utilizaron todas las versiones de cada rollo ya que en estos casos la diferencia de calidad no es tan apreciable.

Todas las hojas de los rollos consisten en imágenes binarias en formato TIFF, todas con una resolución de 3520x4800 píxeles.

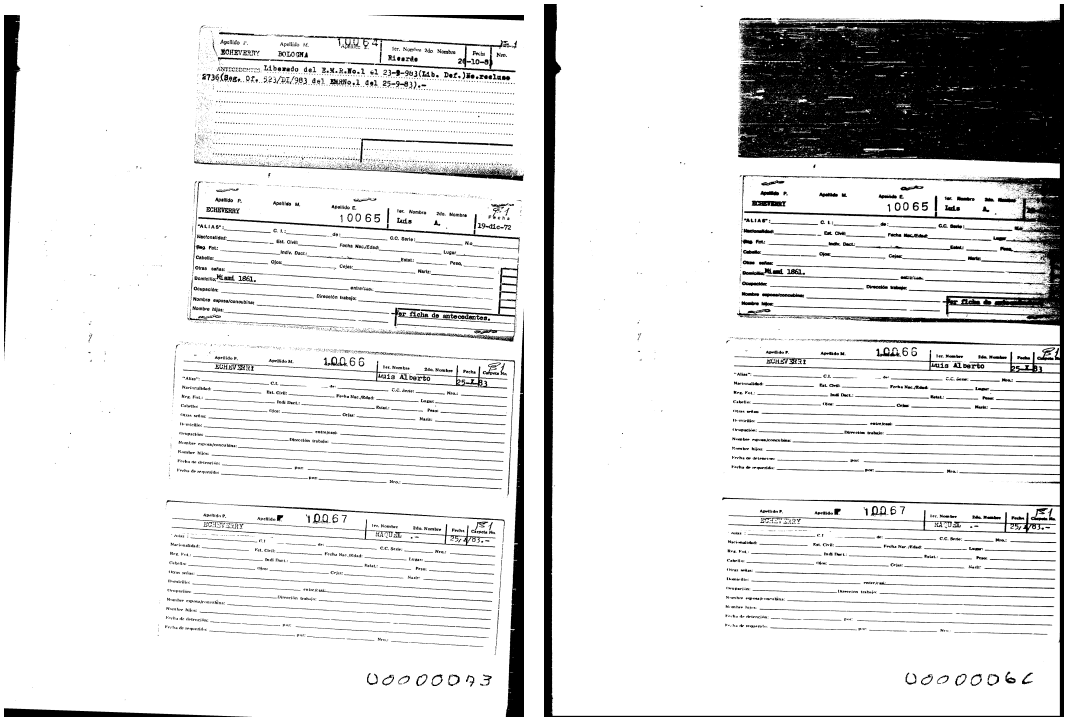


**Figura 3.1:** Hoja 73 del rollo 588, se muestran las dos versiones disponibles de la hoja. La versión de la izquierda se lee bastante más que la de la derecha.

En total se cuenta con 14 rollos (considerando una única versión para cada rollo) los cuales suman 25.474 imágenes que en total contienen fichas de 31,435 individuos diferentes. Se observa que las fichas no se encuentran repartidas de manera uniforme entre todos los rollos. Esto se puede ver en la Tabla 3.1 donde se encuentra un resumen de los rollos pertenecientes al fichero de la O.C.O.A. Dicha tabla presenta información detallada, que incluye el número de rollo, la letra de los apellidos que se encuentran en ese rollo, los números de fichas en ese rollo, el total de fichas en el rollo y la cantidad de imágenes que contiene el rollo.

**Tabla 3.1:** Listado de rollos utilizados.

| # Rollo | Letras | Fichas | # Fichas | # Imágenes |
|---|---|---|---|---|
| 570 | A | 1 a 2.445 | 2445 | 2.169 |
| 584 | B | 2.446 a 5.033 | 2588 | 2.240 |
| 585 | C | 5.034 a 8.216 | 3183 | 2.595 |
| 587 | D | 8.217 a 10.016 | 1800 | 1.435 |
| 588 | E, F | 10.017 a 12.039 | 2023 | 1.627 |
| 594 | G, H | 12.040 a 14.756 | 2717 | 2.105 |
| 599 | I, J, K, L | 14.757 a 17.080 | 2324 | 1.971 |
| 603 | M | 17.081 a 20.173 | 3093 | 2.366 |
| 607 | N, O | 20.174 a 21.288 | 1115 | 864 |
| 608 | P, Q | 21.289 a 23.775 | 2487 | 2.056 |
| 612 | R | 23.776 a 26.071 | 2296 | 1.784 |
| 613 | S | 26.072 a 28.589 | 2518 | 1.931 |
| 614 | T | 28.590 a 29.452 | 863 | 753 |
| 615 | U, V, W, X, Y, Z | 29.453 a 31.435 | 1983 | 1.578 |

El índice está conformado por hojas tipo *fanfold* con renglones. Estas hojas no solo incluyen el nombre de la persona y su número de ficha, sino también la información sobre las organizaciones a las que pertenecía, el número de afiliado dentro de dichas organizaciones, una columna para observaciones y una columna llamada número carpeta que contiene el número de carpeta asociado a esa persona. Esa carpeta no se encuentra dentro de los rollos con los que se va a trabajar. No todas las columnas contienen información en todas las líneas.

En la Tabla 3.2 se muestra la cantidad de imágenes que contienen al índice para todas las versiones de todos los rollos. Como es esperable, a mayor cantidad de fichas en el rollo, el índice tiene más hojas. En total las imágenes del índice son 927.

En cuanto a las fichas individuales, se observa que cada una tiene un número único que se encuentra en todas las partes frontales y partes de atrás que además coincide con el número de ficha del índice. Es por lo tanto, mediante este número que se pueden vincular todas las partes de ficha con el índice para una misma persona. Dicho número es secuencial, comienza en 1 y es único en todo el fichero. Por lo tanto, la ficha número 1 se encuentra en el rollo que contiene las fichas de los apellidos que empiezan con A y la ficha con el número más alto se encuentra en el rollo de los apellidos que empiezan con Z. Por ejemplo, en la ya mencionada Figura

**Tabla 3.2:** Cantidad de hojas del índice de cada rollo.

| # Rollo | # Imágenes índice |
|---|---|
| 570 | 39 |
| 570v1 | 40 |
| 584 | 48 |
| 585 | 55 |
| 585v1 | 55 |
| 587v1 | 32 |
| 588 | 36 |
| 588v1 | 36 |
| 594 | 49 |
| 599 | 43 |
| 599v1 | 41 |
| 599v2 | 43 |
| 603 | 54 |
| 603v1 | 54 |
| 607 | 20 |
| 607v1 | 20 |
| 608 | 44 |
| 608v1 | 42 |
| 612 | 40 |
| 612v1 | 40 |
| 613 | 44 |
| 614 | 15 |
| 615 | 37 |

1.1 los números de ficha de cada una de las partes de ficha en ella son 0307, 0308, 0309 y 0309. El número 0309 se encuentra dos veces ya que son la parte frontal y parte de atrás de la misma ficha. Por lo general, el número se ubica en el tercio superior de la ficha aunque existen excepciones.

Se observa que existen varios tipos de plantillas de partes frontales y partes de atrás. Las plantillas contienen información muy similar, aunque algunas plantillas contienen menos campos. El orden en el que aparecen los campos cambia un poco entre las diferentes plantillas. En términos generales, las fichas se completaron utilizando máquina de escribir aunque también se cuenta con ejemplos completados manualmente o con combinación de texto a máquina y a mano en una misma ficha. Además, todas las diferentes plantillas detectadas de partes de ficha tienen las mismas dimensiones. Después de separar todas las partes de ficha en una misma imagen, se obtiene un total de 61476 partes de ficha.

En la Figura 3.2 se observa la parte frontal de la ficha 95 perteneciente al rollo 570 (letra A). Esta ficha se completó con máquina de escribir y después se agregó nueva información a mano. Además, el campo 'dirección trabajo' fue tachado. Hay campos que no se completaron, entre ellos la cédula de identidad. Se observa también una rotación de unos pocos grados.

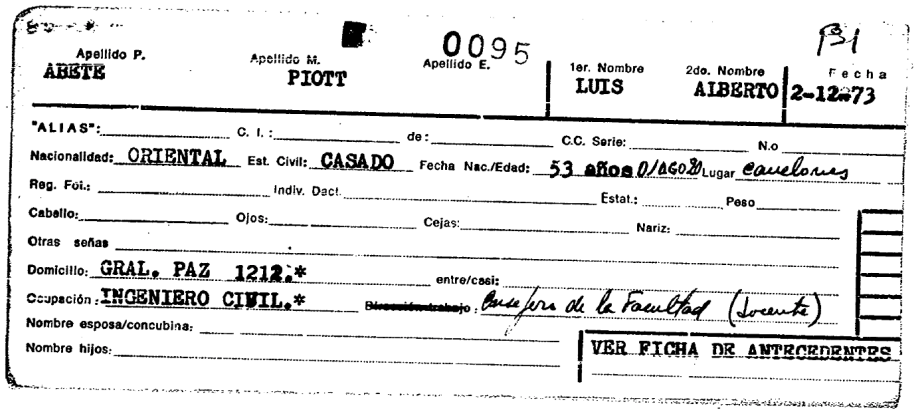



**Figura 3.2:** Ejemplo de parte frontal de ficha, en este caso se completó con máquina de escribir y se agregaron algunos campos a mano.

La parte de atrás de esa misma ficha se encuentra en la Figura 3.3. Se reiteran algunos datos personales como el nombre y el número de ficha (0095). Asimismo, se incluyen los antecedentes de la persona.

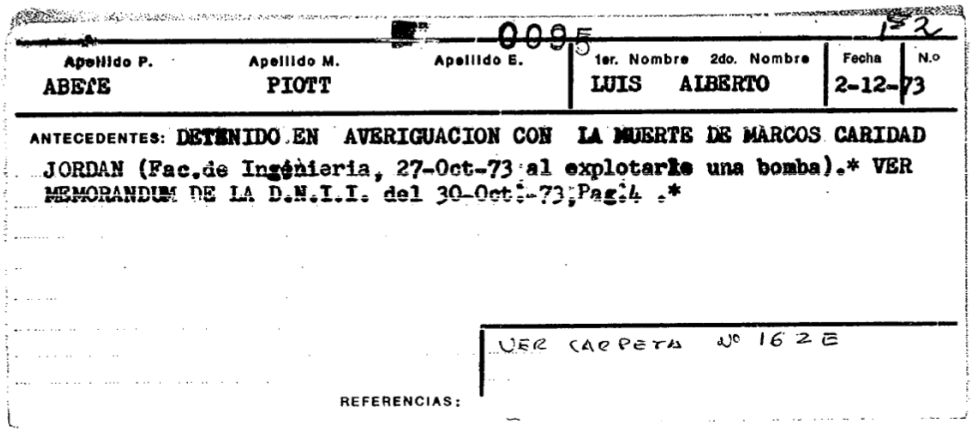



**Figura 3.3:** Ejemplo de parte de atrás de una ficha, contiene antecedentes de la persona.

En general, los campos de la parte frontal se componen de una etiqueta con el nombre del campo y un espacio en blanco para escribir la información de la persona. El listado completo de campos que puede contener una parte frontal es el siguiente: Apellido Esposa, Apellido Paterno, Apellido Materno, Primer Nombre, Segundo Nombre, Fecha, Alias, Cédula (número y departamento en campos diferentes), Credencial (serie y número en campos diferentes), Nacionalidad, Estado Civil, Fecha nacimiento o Edad, Lugar de Nacimiento, Registro Fotográfico, Estatura, Peso, Cabello, Ojos, Cejas, Nariz, Otras señas, Domicilio, Ocupación, Dirección del trabajo, Nombre esposa/concubina, Nombre hijos, Fecha de detención y Fecha de requerido.

Es fundamental destacar que la información relativa a las organizaciones se encuentra exclusivamente en el índice mientras que el nombre se encuentra tanto en el índice como en la ficha. La extracción de información del índice resulta importante ya que permite obtener información no disponible en la ficha, al tiempo que proporciona una segunda fuente para el nombre, lo que genera cierta redundancia en caso de que la ficha presente dificultades de legibilidad en esa parte o tenga algún otro problema.

## 3.1. Problemas de calidad de las fichas

En esta sección se expondrán algunos problemas identificados en las imágenes de las fichas que dificultan el trabajo a realizar. Dichos problemas son clasificados en problemas de adquisición y problemas de calidad de datos. Finalmente, se cierra el capítulo enseñando algunos problemas en las imágenes de los índices.

### 3.1.1. Problemas de adquisición

Los problemas de adquisición se refieren a problemas introducidos durante el proceso de generación de las imágenes. Este proceso se realizó en dos etapas. Primero, los documentos físicos fueron microfilmados y luego se escanearon los microfilms. Se detectan varios problemas de este tipo. El más notorio es que se tienen varias partes de ficha por imagen. Esto genera la dificultad de tener que separar cada parte de ficha en su imagen independiente para poder trabajar con ellas individualmente.

Además, las partes de ficha pueden encontrarse rotadas dentro de una hoja y en ángulos diferentes. Un ejemplo de esta situación se puede observar en la Figura 3.4. Esta figura presenta una hoja que contiene dos partes de ficha, ambas rotadas. Sin embargo, la parte de ficha de arriba a un ángulo mayor. Es importante alinear las partes de ficha ya que en general esto mejora el resultado de los OCR, además de facilitar la extracción de los campos.

En algunos casos, problemas de iluminación durante la adquisición generaron imágenes que se encuentran ilegibles. Este caso se puede ver en la Figura 3.5.

Finalmente, se observan sutiles transformaciones perspectivas. Dadas dos fichas del mismo tipo, alineadas y superpuestas, es posible que la parte común de ambas fichas (la plantilla) no coincida. En la Figura 3.6 se puede ver un ejemplo de dos fichas del mismo tipo que fueron superpuestas y sin embargo la parte común (la plantilla) no coincide en su totalidad. Los campos ‘ALIAS’ y Nacionalidad coinciden pero si se observan campos lejanos como ‘2do. Nombre’ la coincidencia es

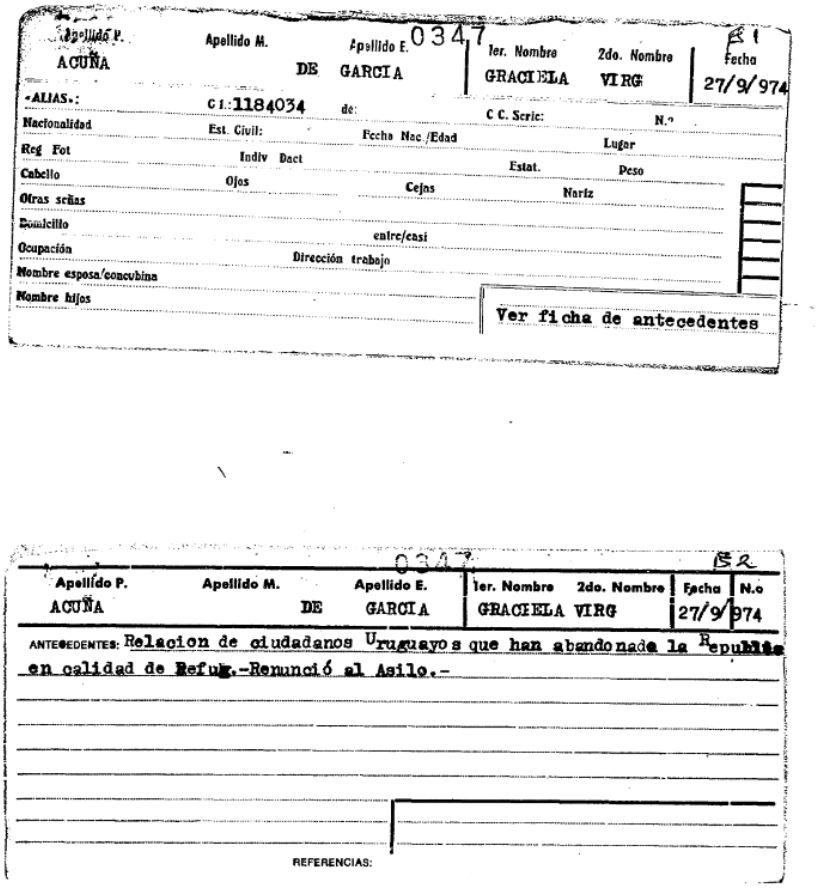



**Figura 3.4:** Hoja 363 del rollo 570. La imagen contiene dos fichas rotadas a diferente ángulo

considerablemente menor.

### 3.1.2. Problemas de calidad datos

A diferencia de los problemas de adquisición, los problemas de calidad de datos se introducen en el momento de completar las partes de ficha. En algunos casos, el texto escrito no está alineado con los renglones de la plantilla. En ocasiones, el texto se encuentra muy por encima del renglón, siendo cortado por el renglón o parte del texto se encuentra por encima del texto que describe el campo. En la Figura 3.7 se encuentra el campo Ocupación de una parte de ficha ejemplificando

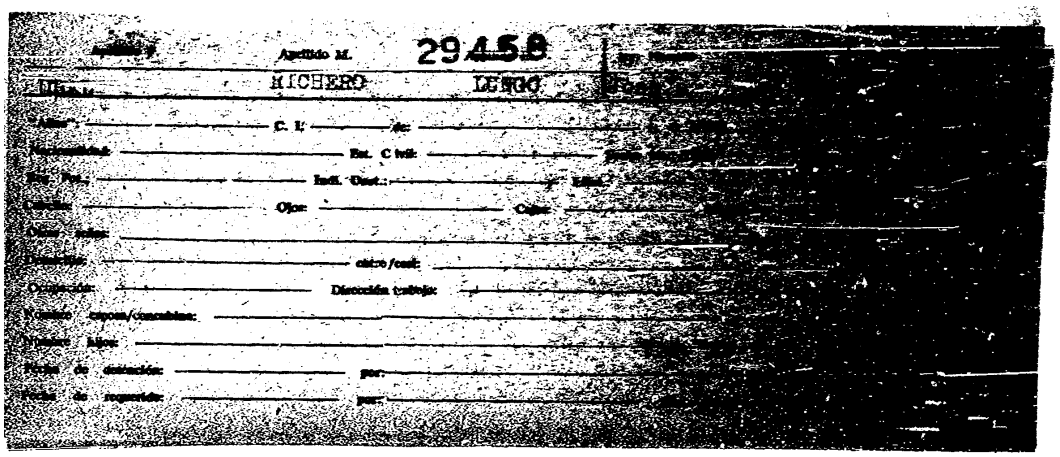



**Figura 3.5:** Ejemplo de una ficha que no puede leerse debido a la iluminación.

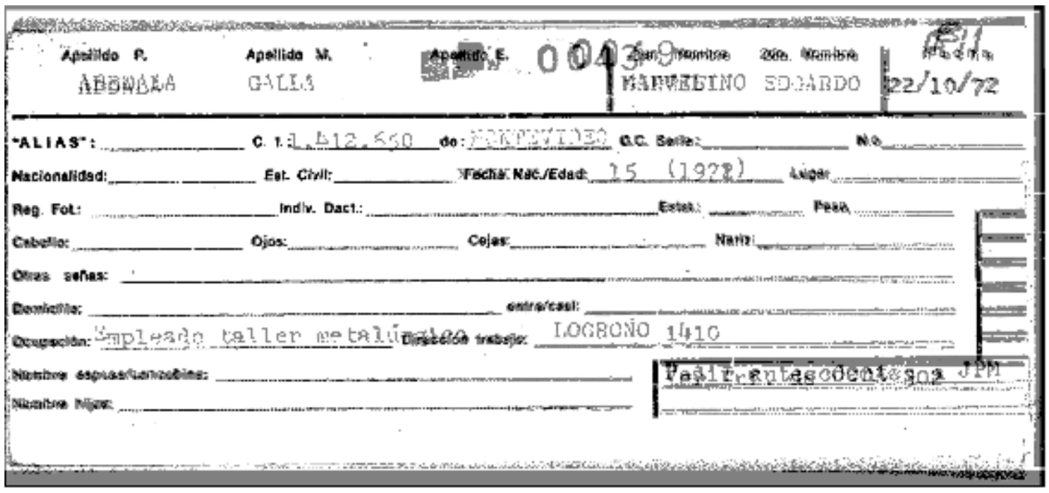



**Figura 3.6:** Ejemplo de dos fichas superpuestas cuya parte común no coincide debido a que hay una transformación perspectiva.

este problema. En este caso particular, el texto se escribió por encima del renglón, quedando desalineado con el texto que describe el campo.

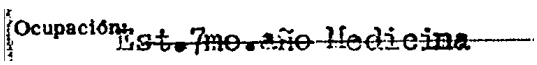



**Figura 3.7:** Ejemplo del campo ocupación con texto cortando el renglón.

Es común que no haya un formato estándar para completar los campos. Por ejemplo, una fecha puede aparecer como ‘10/2/1975’ o como ‘10 feb. 1975’. En la Figura 3.8 se encuentra un ejemplo de este problema, se puede ver el contenido del campo Fecha para tres partes de ficha diferentes. En todos los casos la fecha se encuentra separada por un carácter distinto.

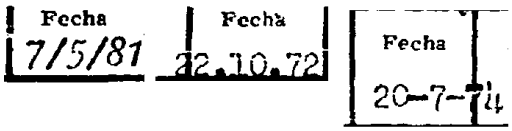



**Figura 3.8:** Varios ejemplos del campo fecha de partes de ficha.

En el caso del campo cédula, en ocasiones los dígitos están separados con puntos y a veces no. Además, en general las fichas no están completadas en su totalidad, quedando campos en blanco en muchos casos.

### 3.1.3. Problemas en los índices

Los índices también presentan algunos problemas de calidad de datos.

Un primer problema observado es que a veces el texto no se encuentra sobre los renglones sino que está un poco por encima o por abajo de ellos. Esto se puede ver en la ya mencionada Figura 1.2.

Además, por problemas de adquisición, en ocasiones los renglones se encuentran parcialmente borrados en partes de la imagen. Asimismo, en casi todas las hojas de los índices se pueden ver marcas de agujeros en los bordes, o marcas que parecerían indicar la forma en que las hojas físicas estaban agrupadas. Estos problemas, una vez más, se presentan en la Figura 1.2.

Otro inconveniente que ocurre en ocasiones es que hay partes que se encuentran completamente borradas. Un ejemplo extremo de este problema se observa en la Figura 3.9. En este caso, casi ningún número se puede leer y hay una gran cantidad de nombres que se encuentran borrados. Como forma de mitigar este problema es que se decide trabajar con todas las diferentes versiones de los índices, de forma que si se encuentra una hoja borrada, existe la posibilidad de que en otra versión de ese índice la hoja sea legible.

Finalmente, cabe mencionar que es posible que el texto en las hojas de los índices se encuentre ligeramente rotado, como sucede con las fichas.

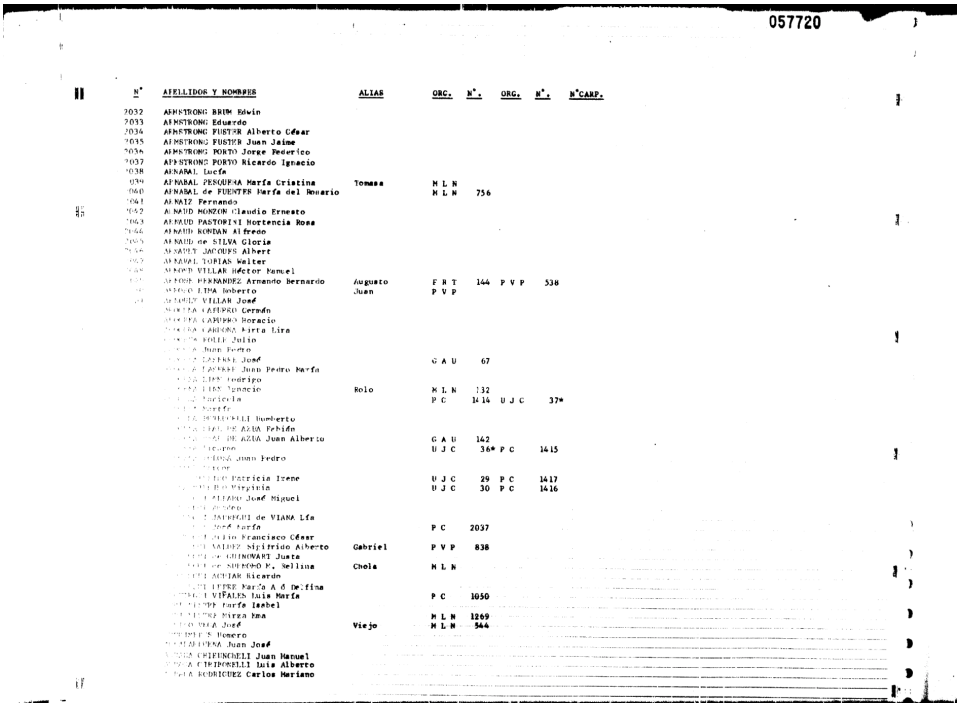



**Figura 3.9:** Ejemplo una hoja del índice perteneciente al rollo 570 parcialmente borrada.

Si bien los datos con los que se va a trabajar cuentan con más problemas de calidad, se expusieron los que representaron una mayor dificultad a la hora de realizar la tesis.

# Capítulo 4

# Metodología

En este capítulo se presenta y describe en detalle la solución implementada para poder extraer la información de las fichas y del índice. El capítulo finaliza con una sección donde se enseña el funcionamiento de la interfaz realizada para facilitar la búsqueda. Para facilitar la explicación, el proceso se va a dividir en etapas. Un diagrama que enseña el nombre de cada etapa y el flujo de resultados intermedios se encuentra en la Figura 4.1.

Se cuenta con dos tipos de hoja diferentes al comienzo del proceso: las hojas que contienen las fichas y las hojas que contienen el índice.

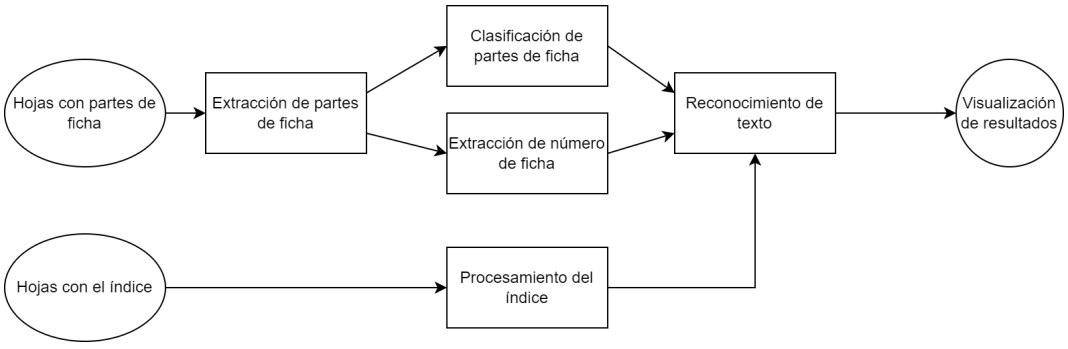



**Figura 4.1:** Diagrama con las diferentes etapas de la metodología seguida.

Recordando del capítulo 1, se le llama parte de ficha a la sección de la hoja que contiene la parte de adelante o de atrás de una ficha. Para las hojas que contienen las

fichas, se comienza por la etapa de extracción de partes de ficha, la cual consiste en separar en varias imágenes las partes de fichas que componen una misma hoja, cada una con una única parte de ficha. Este paso también incluye eliminar espacios en blanco sobrantes a la izquierda y derecha de las partes ficha y enderezar las partes fichas de forma que el texto quede alineado horizontalmente. Dentro de una misma hoja, las partes de ficha pueden estar rotadas con diferentes ángulos.

En la Figura 4.2 se puede ver un ejemplo de una hoja con cuatro partes de ficha a la izquierda y a la derecha el resultado de aplicar el algoritmo de extracción, con las cuatro partes de ficha por separado y enderezadas.

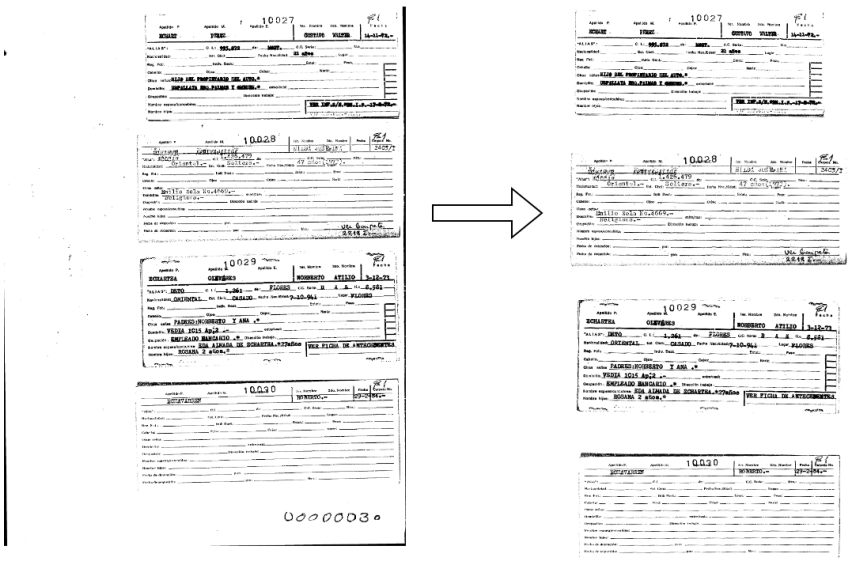


**Figura 4.2:** Ejemplo de ejecutar el algoritmo de separación de partes de ficha en una hoja con tres partes de ficha.

Estas partes de ficha, ahora en imágenes por separado, se utilizan en las etapas de clasificación y extracción de número de ficha.

Existen varias clases de partes de ficha: parte frontal de una ficha, parte de atrás, fotografía, huellas dactilares, etc. Además, dentro de cada clase hay variaciones. Se identifican varios tipos de partes frontales así como varios tipos de partes de atrás. En la etapa de clasificación, se busca detectar si una parte de ficha es una parte frontal o de atrás y en caso de ser parte frontal de qué clase. En la Figura 4.3 se puede ver un ejemplo de una parte de atrás, una parte frontal, una parte de ficha con huellas dactilares y una fotografía.

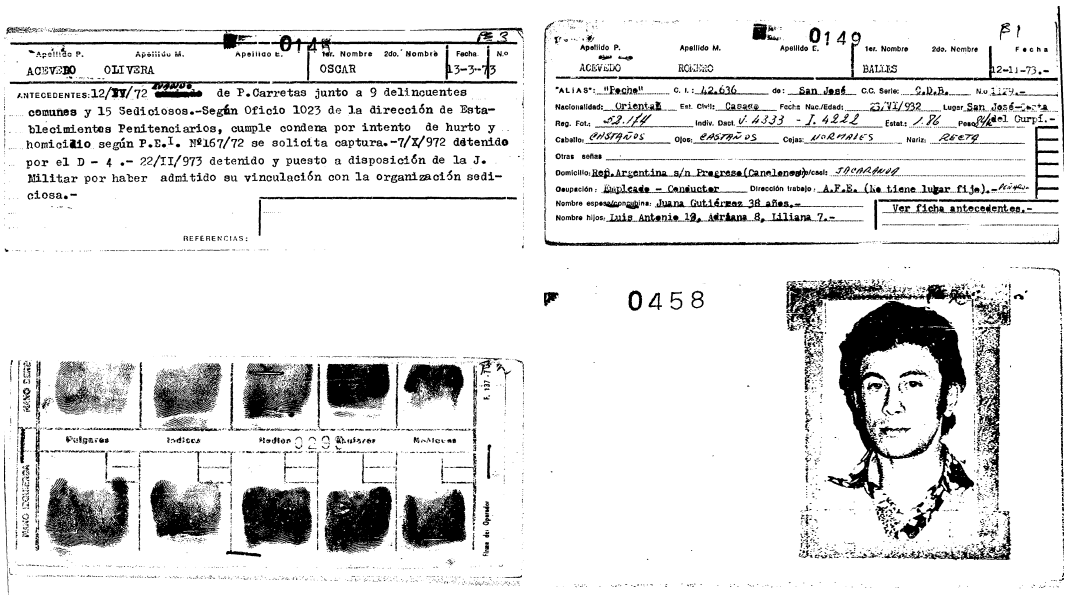


**Figura 4.3:** Ejemplo de clases de parte de ficha detectadas. Arriba a la izquierda se encuentra una parte de atrás. Arriba a la derecha se encuentra una parte frontal. En las imágenes de abajo, a la izquierda hay huellas dactilares y a la derecha una fotografía.

Durante la etapa de extracción de número de ficha, se busca para cada imagen extraer el número de ficha. Este número es importante para poder asociar todas las partes frontales y de atrás junto con el índice a una misma persona.

En esta misma etapa, para cada parte frontal se busca extraer la información escrita en los diferentes campos para luego aplicar técnicas para convertir la imagen a texto. Esta información debe almacenarse en una base de datos.

Durante la etapa de procesamiento del índice, se trabaja sobre el índice de cada letra, separando las líneas y las columnas, convirtiendo las imágenes extraídas a texto.

En este momento, el resultado de procesar el índice, los números de ficha y los campos extraídos de las partes frontales son la entrada de la etapa de reconocimiento de texto, que como su nombre lo indica, busca reconocer el texto en las imágenes. Por ejemplo, en el caso de la hoja que se encuentra en la Figura 1.1, la entrada a la etapa de reconocimiento de texto va a ser el contenido de los campos de las dos partes frontales de ficha e imágenes con los números de ficha ‘0308’ y ‘0309’. En la Figura 1.2, la entrada de esta etapa son las imágenes que contienen las celdas de

la tabla.

Finalmente, se desarrolla una interfaz gráfica para poder acceder y navegar fácilmente por la información extraída y para visualizar los resultados.

Para cada una de las etapas mencionada, hay una sección asociada en este capítulo que explica en detalle el trabajo realizado. Finalmente, el capítulo concluye con una descripción de la aplicación desarrollada para poder visualizar los resultados.

## 4.1. Extracción de partes de fichas de las hojas

Esta etapa del proceso tiene como entrada las imágenes de los rollos con múltiples partes de ficha, separa cada una de las partes de ficha en una imagen individual y almacena cada una de ellas en su propia imagen. En la Figura 4.4 se encuentra una imagen de ejemplo que ilustra el proceso.

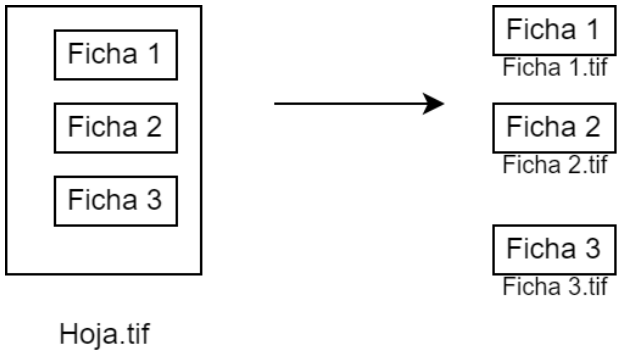



**Figura 4.4:** Ilustración de la etapa de extracción de partes de ficha.

Para explicar el funcionamiento del algoritmo de separación, se decide desglosar el mismo en varias etapas. La primera etapa consiste en un preprocesamiento de la imagen, esta es una etapa de preparación de la imagen. Una vez la imagen fue preprocesada, se pasa a la etapa de determinar la cantidad de partes de ficha en la imagen. Luego de determinada esta cantidad, se obtienen los índices (es decir la posición en píxeles en sentido vertical) de comienzo y fin de cada parte de ficha en la hoja y se extraen las partes de ficha. En este momento, se trabaja individualmente con cada parte de ficha extraída. Se enderezan para que el texto quede derecho, se

realiza un ajuste de los índices detectados previamente con la parte de ficha derecha para mejorar el recorte, se elimina espacio sobrante a los costados de la parte de ficha y se guarda en su propia imagen.

Para todo el proceso de separar las partes de fichas se va a hacer mucho uso del vector de la suma de intensidades, tanto por filas como por columnas. Este vector se utiliza durante el proceso de determinar la cantidad de partes de ficha, durante la extracción y para el algoritmo de enderezar. A este vector también le llamamos vector C para simplificar. Dada una imagen $I$ con $i$ filas y $j$ columnas, la suma de intensidades por columnas se define de la siguiente manera:

$$C[z] = \sum_{k=0}^{i-1} I(k, z)$$

La suma de intensidades por filas se define de la siguiente manera:

$$C[z] = \sum_{k=0}^{j-1} I(z, k)$$

Es decir, el cálculo de la suma de intensidades consiste en sumar las filas o columnas de la imagen. Con el uso de este vector, se puede calcular en qué posición se encuentran las partes de ficha dentro de la imagen. En la Figura 4.5 se encuentra el vector C por filas de una imagen que tiene tres partes de ficha. La imagen completa junto con el vector C en una misma gráfica se encuentran en la Figura 4.6.

Se puede ver que entre las posiciones 1000 y 2000 y las posiciones 2700 y 3500 hay valores muy pequeños, esto se corresponde con los espacios en blanco donde no hay partes de ficha en la imagen y se puede verificar en la Figura 4.6. La idea detrás del algoritmo de separación de partes de fichas es lograr detectar en el vector de la suma de intensidades dónde se encuentran las partes de ficha aprovechando que los valores más bajos se dan cuando hay un espacio en blanco.

El algoritmo desarrollado está relacionado con el trabajo de Pavlidis y Zhou, 1992 mencionado en la sección 2.1.1. Sin embargo, en este caso la tarea es más

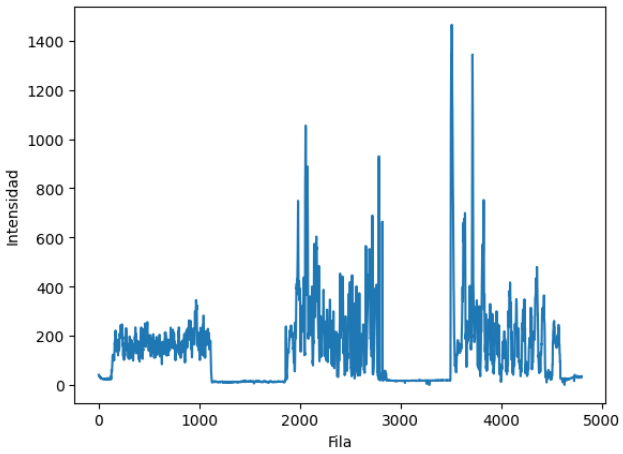



**Figura 4.5:** Suma de intensidades por fila de la hoja 218 del rollo 570. Los colores se encuentran invertidos para que mayor intensidad represente mayor cantidad de píxeles negros en la fila.

sencilla. Se conoce que la cantidad de partes de ficha es entre una y cinco (mencionado en el capítulo 1). Además, la altura es fija y están posicionadas en una sola columna.

### 4.1.1. Preprocesamiento de imagen

Durante este paso, se realiza un preprocesamiento de la imagen. Se invierten los colores y se normaliza la imagen para que en donde haya texto (color negro) el valor sea $1$ y donde hay espacios en blanco el valor sea $0$. De esta manera, al calcular el vector de intensidad por filas se obtiene un valor alto si esa fila contiene una parte de ficha y un valor bajo si hay un espacio en blanco.

Las imágenes pueden contener bordes negros, que agregan ruido. Para mitigar este problema, se eliminan 300 píxeles en cada lado de la imagen (solo en ancho). Este número se obtiene mediante observación de las imágenes. A este margen se le llama margen de seguridad de ruido.

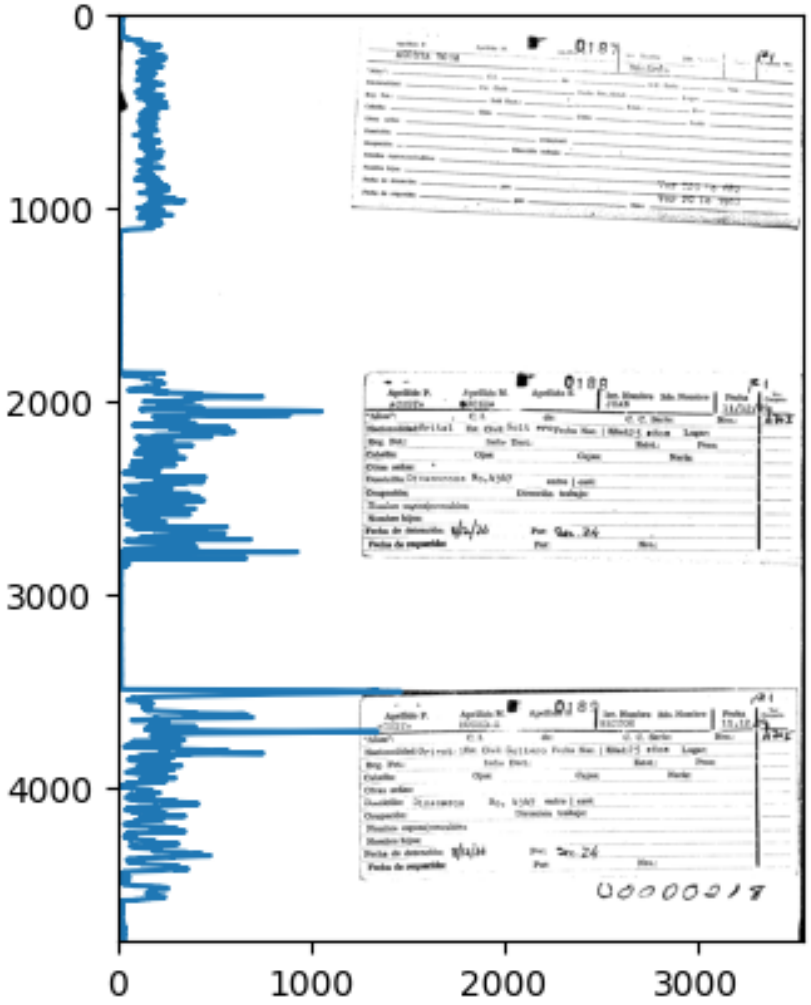



**Figura 4.6:** Suma de intensidades por fila de la hoja 218 del rollo 570 junto con la imagen original.

### 4.1.2. Determinar la cantidad de partes de ficha en la imagen

En el capítulo 1 se menciona que la cantidad de partes de ficha en cada hoja es entre una y cinco. En esta etapa del proceso del algoritmo de extracción, se determina la cantidad de partes de ficha en la hoja que está siendo procesada. La intuición detrás de este paso es comenzar utilizando el vector de la suma de intensidades por fila de la hoja, que va a contener muchos picos, y aplicar varias operaciones de filtrado de forma que el valor del vector sea 1 en las posiciones donde hay parte de ficha, y 0 donde no hay. De esta manera, dividiendo la suma del vector resultante entre la altura de una parte de ficha, se obtiene la cantidad aproximada de partes de

ficha en la hoja.

Para comenzar, se considera una versión aún más reducida de la imagen, eliminando 920 píxeles de cada lado de la imagen, lo que da un total de 1220 píxeles eliminados, al sumar los del paso anterior. De esta manera, se obtiene una imagen donde ninguna parte de ficha tiene espacios en blanco a la derecha o a la izquierda, aunque probablemente las partes de ficha queden cortadas. Por ejemplo, en la Figura 4.7 se encuentra el resultado de esta operación para la hoja 287 del rollo 570, donde Hay 3 partes de ficha y todas quedan cortadas.

**Figura 4.7:** Hoja 287 del rollo 570 con 1220 píxeles recortados de cada lado. Las partes de ficha fueron recortadas casi a la mitad. Originalmente, se encontraban sobre el lado derecho de la hoja.

Ahora, se calcula sobre esta nueva imagen el vector de intensidades por fila y se aplica una umbralización con un umbral $T = 50$. Es decir, si para una fila la suma es mayor o igual a $T$, se convierte en el valor $1$ y si es menor entonces se convierte en $0$. Este valor fue hallado mediante ensayo y error, analizando el resultado obtenido para varias imágenes.

En este momento, el vector resultante va a ser igual a $1$ donde haya parte de ficha o borde de parte de ficha y $0$ donde no haya parte de ficha o entre los renglones de una parte de ficha. El objetivo siguiente es que solo haya $0$ en los espacios entre partes de ficha.

Esto se logra aplicando una dilatación morfológica sobre el vector C con un elemento estructurante de tamaño 54 (también obtenido mediante ensayo y error para varias imágenes).

La dilatación morfológica es una operación que busca expandir estructuras existentes en un vector. En este caso, se utiliza como forma de rellenar posibles huecos que queden en el vector C. Dado un vector C binario y un elemento estructurante de tamaño $n$, se define la dilatación morfológica ($dm$) para la componente $j$ en la Ecuación 4.1. Es decir, la posición $j$ del vector resultante contiene $1$ solamente si hay alguna posición vecina a distancia menor o igual a $n$ que contenga $1$.

$$dm(C,n)_j = \begin{cases} 1 & \text{si } \sum_{i=j-n}^{j+n} C(i) > 0 \\ 0 & \text{en otro caso} \end{cases} \tag{4.1}$$

En consecuencia, si se tiene una imagen con cuatro partes de ficha correctamente separadas, el resultado de aplicar la dilatación morfológica es un vector cuya gráfica muestra cuatro pulsos cuadrados (cuya altura es $1$) aproximadamente de largo equivalente a la altura de una parte de ficha (se mencionó anteriormente que todas las partes de ficha tienen las mismas dimensiones al comienzo del capítulo 3). Sin embargo, debido a que hay muchos casos donde las partes de ficha no están correctamente separadas es posible que no haya tantos pulsos como partes de ficha. No obstante, en estos casos se va a tener menos pulsos pero más grandes. Por ende, la solución consiste en sumar el vector dilatado, dividir la suma entre la altura de una parte de ficha y redondear ese número. Este número nos da un resultado aproximado de la cantidad de partes de ficha en la imagen.

En la Figura 4.8 se encuentran las sucesivas transformaciones por las que va pasando el vector de la suma de intensidades por fila de la hoja para determinar la cantidad de partes de ficha. La gráfica superior izquierda contiene el vector de

suma de intensidades calculado luego de cortar el margen de los bordes. La imagen superior derecha contiene el resultado de la umbralización. Finalmente, en la parte inferior izquierda se encuentra el resultado de aplicar la dilatación morfológica, se observan tres pulsos, del mismo tamaño lo que nos indica que hay tres partes de ficha en esa hoja.

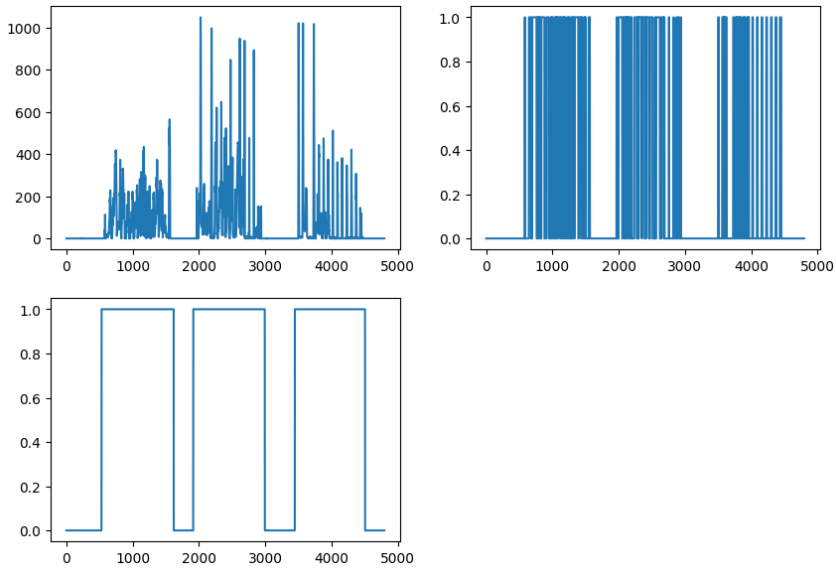



**Figura 4.8:** Vectores C de la hoja 287 del rollo 570 luego de ir aplicando las transformaciones para determinar la cantidad de partes de ficha.

### 4.1.3. Extraer las partes de ficha de la imagen

Conociendo la cantidad de partes de ficha en la imagen, se procede a hallar los índices donde empieza y termina cada parte de ficha.

Para esto se plantea un problema de optimización: dada la cantidad $x$ de partes de ficha detectadas en la imagen, se colocan $x$ ventanas en el vector $C$ de la suma de intensidades por fila y se busca posicionar las ventanas de forma de maximizar la suma dentro de las ventanas. Cada ventana tiene de tamaño la altura de una parte de ficha. El problema se resuelve probando todas las combinaciones posibles.

Este problema es bastante costoso computacionalmente, ya que al aumentar la cantidad de partes de ficha en la hoja aumentan la cantidad de combinaciones. Sin

embargo, hay que tener en cuenta que luego de posicionada una ventana, la siguiente ventana debe encontrarse abajo de ella y no pueden intersectarse. Como consecuencia, al aumentar la cantidad de partes de ficha, se reduce la cantidad de posiciones válidas para las ventanas anteriores.

Además, en lugar de mover las ventanas de a un píxel, se toma un paso más grande y se mueven de a 5 píxeles, reduciendo aún más las posibilidades.

En el capítulo 3, se menciona que la altura de la hoja es de 4800 píxeles. La altura de una parte de ficha es de 959 píxeles. En el caso de tener una sola parte de ficha, al utilizar un paso de 5 se reduce la cantidad de posiciones de 3841 a solo 769. Este aumento del tamaño del paso tampoco afecta en gran medida el resultado ya que 5 píxeles es menos de un 1 % de la altura de la parte de ficha. Como resultado, todo el algoritmo de extracción logra resolverse en un máximo de 45 segundos en el peor caso. Considerando un equipo con 40 núcleos disponible en Cluster.uy, que se cuenta con 25.474 imágenes en total (mencionado en el Capítulo 3), asumiendo el peor caso para todas las imágenes y ejecutando el algoritmo en paralelo (una imagen en cada núcleo), se demoraría aproximadamente 8 horas procesar todas las imágenes.

En la Figura 4.9 se puede ver un ejemplo de una hoja del rollo 570 que contiene tres partes de ficha junto con el vector de la suma de intensidades por fila graficado al costado. La primera parte de ficha corresponde a la parte frontal de la ficha 0269 y las dos restantes son partes de atrás de esa misma ficha. La ficha 0269 es un ejemplo de ficha con una parte frontal y dos partes de atrás.

Para hallar los índices de comienzo y fin de cada parte de ficha, se colocan tres ventanas y se busca que la suma del vector C dentro de las ventanas sea máxima. En la Figura 4.10 se encuentra la misma hoja, con el vector C y sombreadas las ventanas que hacen que la suma del vector C sea máxima.

Una vez se obtuvieron las ventanas de intensidad máxima, se recorta la imagen en esos mismos índices y se logra obtener cada parte de ficha por separado.


**Figura 4.9:** Hoja 287 del rollo 570 junto con el vector de la suma de intensidades graficado al costado.

### 4.1.4. Enderezar las partes de ficha

Una vez obtenidas las partes de ficha, es necesario enderezarlas. Para lograr enderezar las partes de ficha, se necesita conocer el ángulo que hace que el texto quede alineado horizontalmente. A este ángulo lo llamamos 'ángulo óptimo'.

Se decide seguir la metodología de alineación por proyección de perfil explicada en la sección 2.3.1 ya que es el método más sencillo que se puede utilizar para enderezar texto. La medida utilizada para calcular qué tan derecha se encuentra una parte de ficha es la variación total (TV). Dado un vector $v$ de largo $n$ se define la

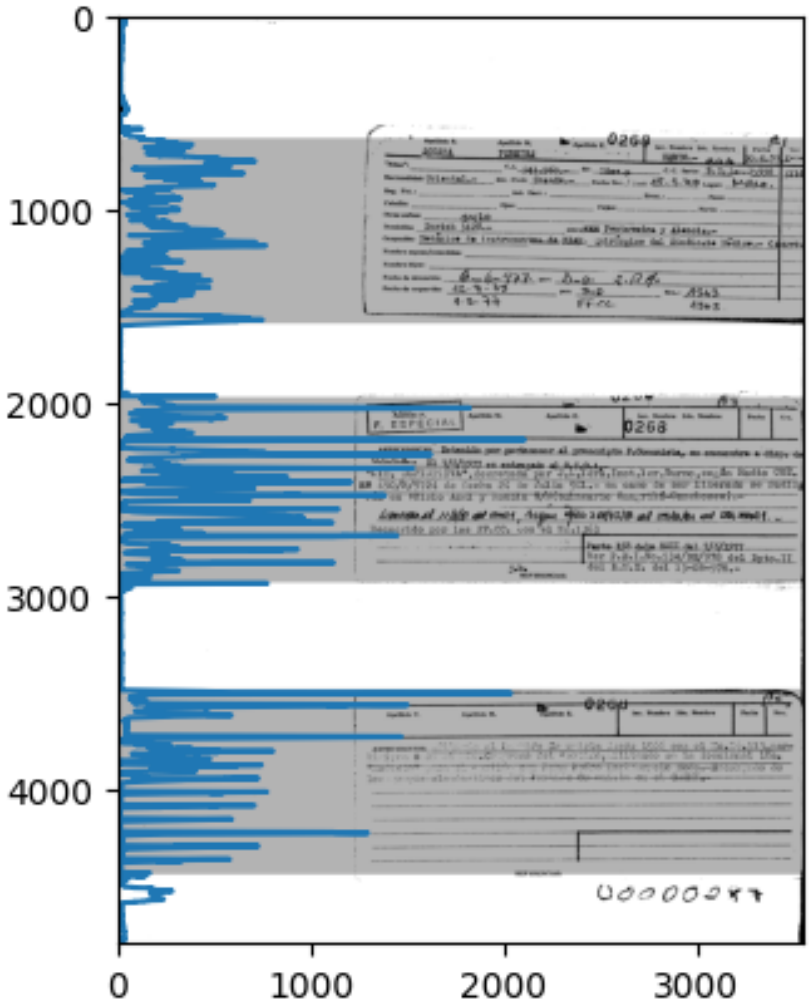



**Figura 4.10:** Hoja 287 del rollo 570 junto con el vector de la suma de intensidades graficado al costado y las ventanas en donde se da el máximo marcadas.

variación total L1 (TV-L1) de la siguiente forma:

$$tvl1(v) = \sum_{i=0}^{n-1} |v[i] - v[i+1]|$$

Es decir, la variación total de un vector es la suma de todas las diferencias absolutas entre dos valores contiguos del vector. Si consideramos el vector de la suma de intensidades por fila y calculamos su variación total, cuánto más grande sea, entonces más derecha se encuentra la parte de ficha.

Si se considera una parte de ficha alineada, entonces el vector de la suma de intensidades por fila va a contener pulsos cuadrados donde haya líneas de texto. Estos pulsos cuadrados generan que la imagen tenga TV-L1 máxima. Los únicos lugares que suman a la variación total son los índices dónde comienza y termina una línea. El valor que aportan es exactamente la suma del vector de la suma de intensidades por fila ya que el otro punto es $0$. Si la imagen se encontrara rotada, si bien hay más puntos que suman a la variación total, la variación en cada uno de ellos es pequeña y, por lo tanto, nunca va a alcanzar al caso que se da cuando la imagen está derecha.

Esto aplica para el caso en el que se representa blanco como $0$ y negro (es decir texto) como $1$. En caso de representar blanco como $1$ y negro como $0$, entonces se debe buscar el mínimo.

En la Figura 4.11 se encuentra una parte de ficha derecha junto con el vector de suma de intensidades por fila. En este caso al calcular el vector se considera blanco como $0$ y negro como $1$. La variación total en este caso es 30154. La misma parte de ficha pero rotada 15 grados se encuentra en la Figura 4.12. La variación total es de 7540, mucho menos que cuando la parte de ficha se encuentra derecha. Además, si se compara la forma de los vectores de suma de intensidades, cuando la parte de ficha está derecha el vector tiene picos más marcados y alcanzan valores más altos.

El algoritmo de enderezado implementado es el siguiente: cada una de las partes de ficha extraídas de la imagen se rota por varios ángulos y para cada ángulo se calcula la variación total del vector C. El ángulo que maximiza la variación total es el ángulo óptimo para esa parte de ficha. Este ángulo óptimo hace que la parte de ficha se encuentre derecha.

Se probaron ángulos en el rango de -8º a 8º con un paso de $0,1$. Este conjunto fue hallado observando partes de ficha. Si se considera este conjunto como:

$$A = \{-8, -7.9, ..., 7.9, 8\}$$

Entonces, el ángulo óptimo es:

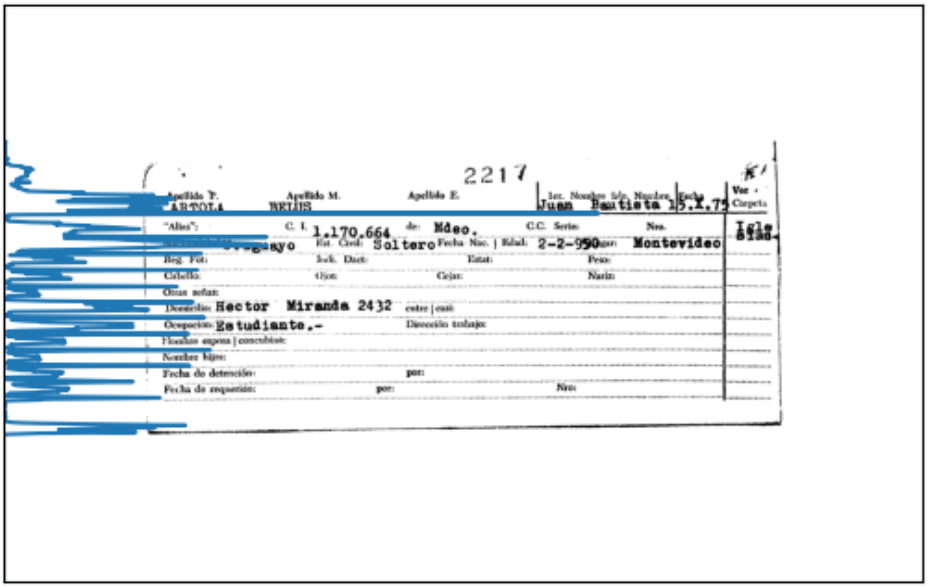



**Figura 4.11:** Ejemplo de una parte de ficha derecha junto con la suma de intensidades por fila.

$$\max_{a \in A} tvl1(C\_rotated\_a)$$

Dónde $C\_rotated\_a$ es la suma de intensidades por fila de la parte de ficha rotada por el ángulo $a$.

Se decide usar como centro de rotación el centro de la parte de ficha ya que este es el centro de la información que se desea enderezar y también resulta más intuitivo que utilizar el borde del recorte. Lamentablemente, debido a que la ficha puede estar posicionada a la izquierda, en el centro o a la derecha de la hoja, no es inmediato calcular este punto. Como se tiene la parte de ficha recortada verticalmente, se puede asumir que a la mitad de la altura se encuentra también la mitad de la altura del recorte. Resta entonces calcular el centro de la ficha en el otro eje. En la Figura 4.13 se encuentra una parte de ficha recortada verticalmente mediante el algoritmo, la parte de ficha se encuentra del lado derecho y se puede ver un poco de ruido sobre el borde izquierdo. En este caso, la mitad de la altura coincide aproximadamente con la mitad de la parte de ficha.

Para obtener el centro de rotación en el eje horizontal, se opta por calcular el

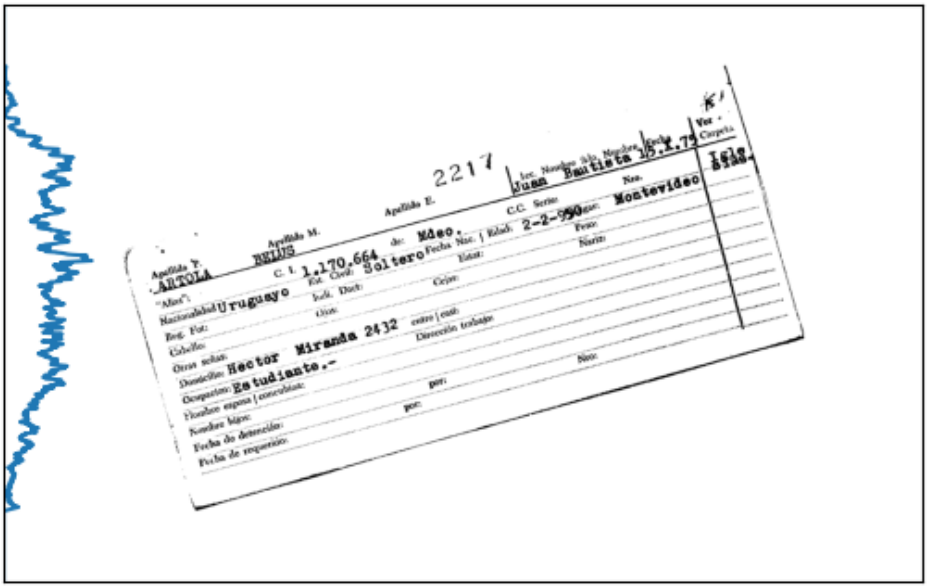


**Figura 4.12:** Ejemplo de una parte de ficha rotada 15 grados junto con la suma de intensidades por fila.

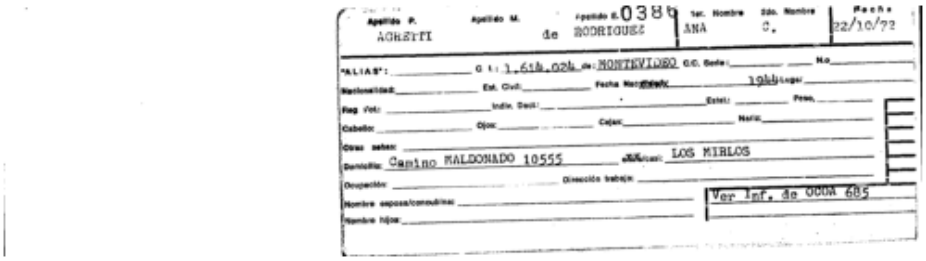



**Figura 4.13:** Segunda parte de ficha de la hoja 401 del rollo 570. Sin recortar los bordes y sin enderezar

baricentro del vector de la suma de intensidades por columnas. Es posible que los espacios en blanco en la ficha muevan el baricentro por lo que antes se aplica una umbralización y una dilatación morfológica para rellenar posibles huecos. Con este valor, se logra aproximar el centro de la parte de ficha. En la Figura 4.14 se encuentra el vector C por columnas para la ficha en la Figura 4.13. La umbralización del vector se encuentra en la Figura 4.15 y finalmente el vector umbralizado y dilatado en la Figura 4.16.

En la Figura 4.17 se puede observar el resultado de obtener el centro de rotación mediante este método, para la parte frontal que se encuentra en la Figura 4.13. Se aproxima bastante al centro de la parte de ficha.

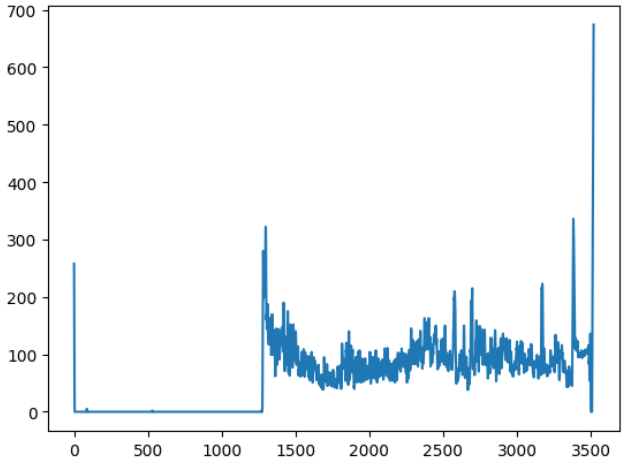
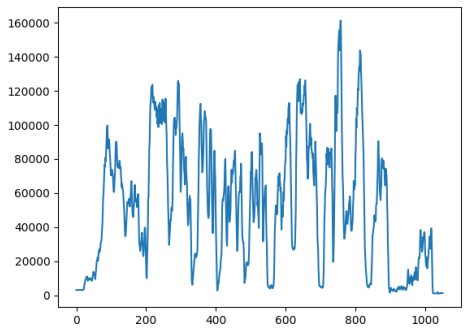
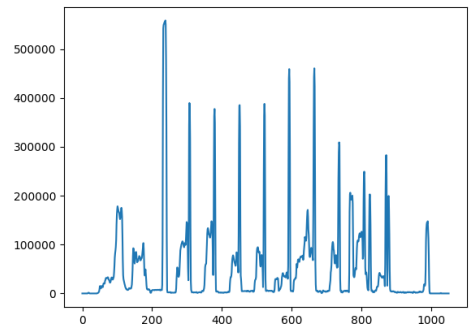



**Figura 4.14:** Suma de intensidades por columna de la segunda parte de ficha de la hoja 401 del rollo 570.

En la Figura 4.18, se encuentra el vector de la suma de intensidades por fila para la segunda parte de ficha de la hoja 401 del rollo 570 antes de enderezar a la izquierda y luego de enderezar a la derecha. Se puede observar que la gráfica de la suma de intensidades por fila de la parte de ficha enderezada tiene picos mucho más marcados que antes de rotar.




**(a)** Suma de intensidades por fila para la segunda parte de ficha de la hoja 401 del rollo 570 antes de enderezar.




**(b)** Suma de intensidades por fila para la segunda parte de ficha de la hoja 401 del rollo 570 luego de enderezar.

**Figura 4.18:** Suma de intensidades por fila para la segunda parte de ficha de la hoja 401 del rollo 570 antes y después de enderezar.

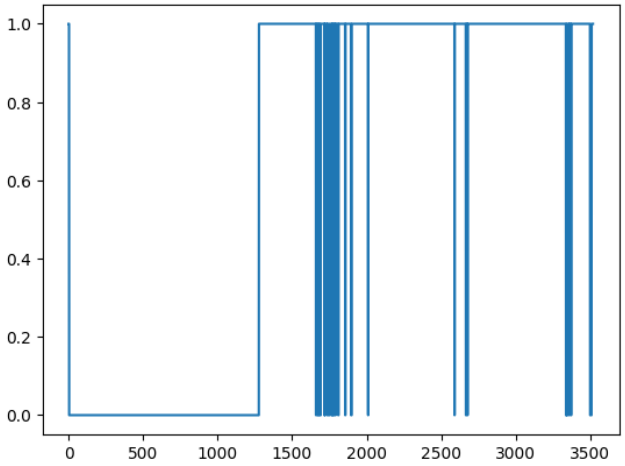



**Figura 4.15:** Suma de intensidades de la segunda parte de ficha de la hoja 401 del rollo 570 umbralizada.

### 4.1.5. Ajustar índices usando el ángulo óptimo

Si bien las partes de ficha tienen todas la misma altura, al encontrarse rotadas debe considerarse una altura un poco mayor (dependiendo del ángulo de rotación) de forma de no dejar ninguna sección afuera del recorte. Con el ángulo óptimo de cada parte de ficha conocido, se puede calcular la altura extra y extraer las partes de fichas en su totalidad.

Por ejemplo, en la Figura 4.13 que contiene la segunda parte de ficha de la hoja 401 del rollo 570 se puede ver que el número de ficha queda un poco recortado. El ángulo óptimo de esta parte de ficha es -1,3º.

Para calcular la altura extra, para cada parte de ficha se multiplica el seno del ángulo óptimo por la altura de la parte de ficha y se toma el valor absoluto. Ahora, se repite de nuevo el paso de encontrar los índices y extraer las partes de ficha. Esta vez, a las ventanas se les suman las alturas extra calculadas para aumentar su tamaño.

Utilizando el ángulo óptimo ya conocido, se enderezan nuevamente las partes de ficha extraídas.

Para la segunda parte ficha de la hoja 401 del rollo 570, la altura extra resultante

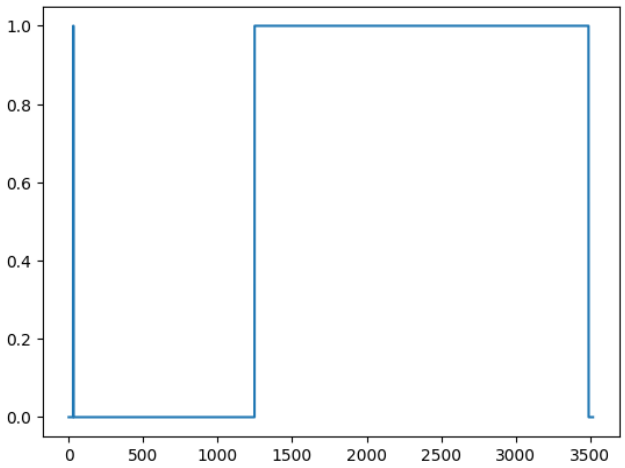



**Figura 4.16:** Suma de intensidades por columna de la segunda parte de ficha de la hoja 401 del rollo 570 umbralizada y dilatada.

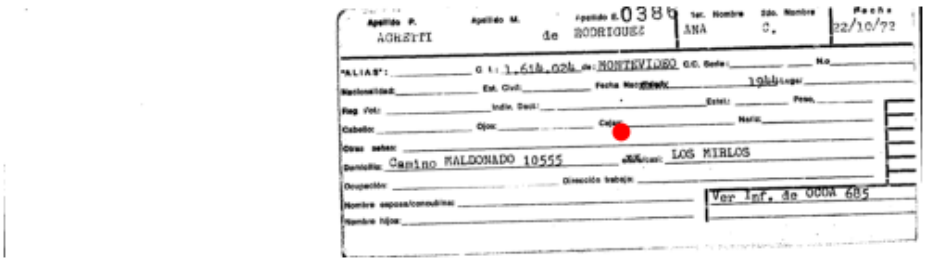



**Figura 4.17:** Segunda parte de ficha de la hoja 401 del rollo 570. Sin recortar los bordes y sin enderezar con el centro de rotación calculado.

es de 53. En la Figura 4.19 se encuentra la parte de ficha recortada nuevamente considerando la altura extra. Esta vez, el número de ficha se encuentra entero.

Es importante mencionar que si bien se busca el ángulo óptimo y altura extra utilizando una imagen que se le recortaron los bordes, una vez obtenido el ángulo óptimo las partes de ficha se extraen sobre la imagen original, de ancho completo.

En las imágenes con cinco partes de ficha, la separación es muy pequeña o inexistente, como muestra la Figura 4.20. En estos casos este paso no se realiza ya que no se tiene espacio hacia dónde ajustar los índices y se corre el riesgo de terminar cortando secciones de otras fichas en la hoja.

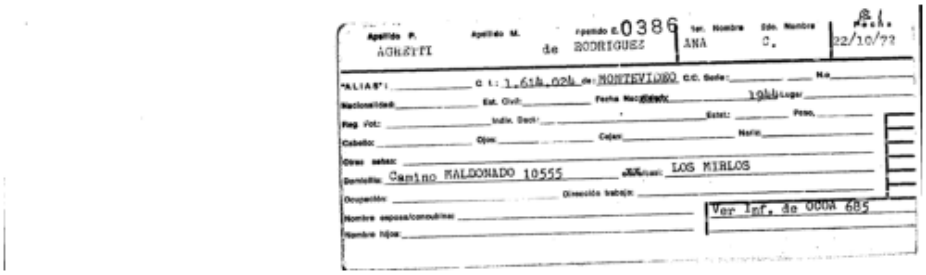



**Figura 4.19:** Segunda parte de ficha de la hoja 401 del rollo 570, recortada con su altura extra.

### 4.1.6. Eliminar espacios verticales

El último paso antes de guardar las imágenes es recortar el espacio sobrante a la derecha o a la izquierda de la parte de ficha, ya que en este espacio no se encuentra ninguna información que se quiera extraer. Eliminar este espacio sobrante va a reducir el tiempo de ejecución a la hora de detectar la ubicación de los campos. Como la imagen resultante es más pequeña, el espacio de búsqueda se ve reducido. Las partes de ficha pueden estar posicionadas a la izquierda, en el centro o a la derecha en la hoja como se vio en los ejemplos de los capítulos anteriores.

Para esta tarea se utiliza el vector de la suma de intensidades, en esta ocasión por columna en lugar de por fila. Una vez calculado este vector, se aplica una umbralización con el objetivo de obtener $1$ donde hay parte de ficha y $0$ donde no la hay.

En la Figura 4.21 se puede ver una parte de ficha con el vector C por columnas graficado por encima. Se puede ver que tiene valores altos donde hay parte de ficha y valores muy bajos donde hay espacio en blanco. Además, hay ruido en el borde izquierdo lo que causa un pico.

Debido a la existencia de ruido en donde se encuentra la parte de ficha, es posible que queden agujeros en el vector. La solución que se aplica para este problema es rellenar los huecos que hayan quedado. En caso de detectar un hueco de tamaño igual o menor a $200$ píxeles, se lo rellena. Se utiliza la función *remove_small_holes* de la biblioteca *skimage* con este fin.

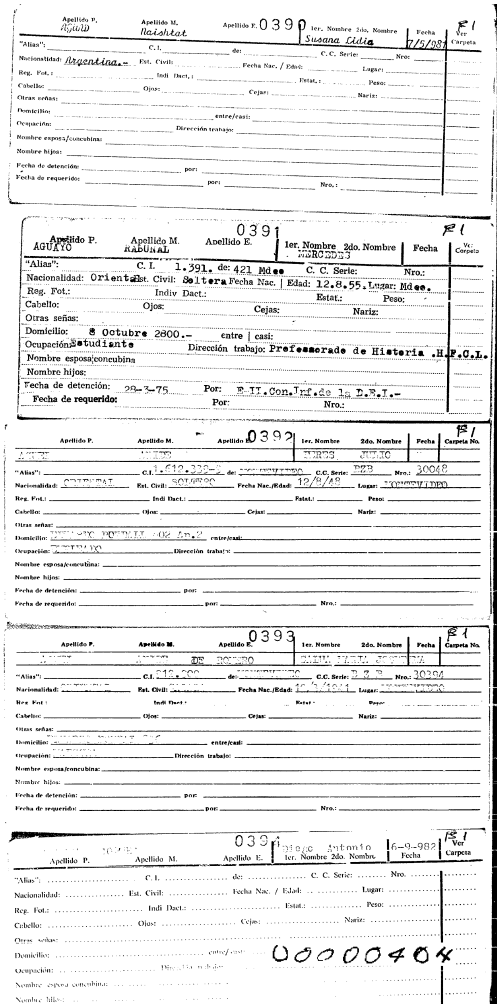


**Figura 4.20:** Hoja 404 del rollo 570.

También es posible que haya ruido en los bordes de la imagen, lo que provoca tener $1$ en los bordes sin que haya parte de ficha. Para solucionar este problema, se define un margen de $300$ píxeles. Luego, se toma el vector umbralizado con agujeros rellenados y se suma tanto para el principio como para el final una cantidad de píxeles igual al margen.

Si para alguno de los lados el resultado de la suma es menor al margen, entonces esos valores se convierten en $0$ para ese lado. En caso contrario, no se realiza ninguna acción. La idea detrás de esto es que si alguno de los bordes tiene ruido, el ruido luego de rellenar agujeros en el vector va a ser de menor tamaño que el margen. Por lo tanto, para que esto funcione, el tamaño del ruido debe ser menor al margen.

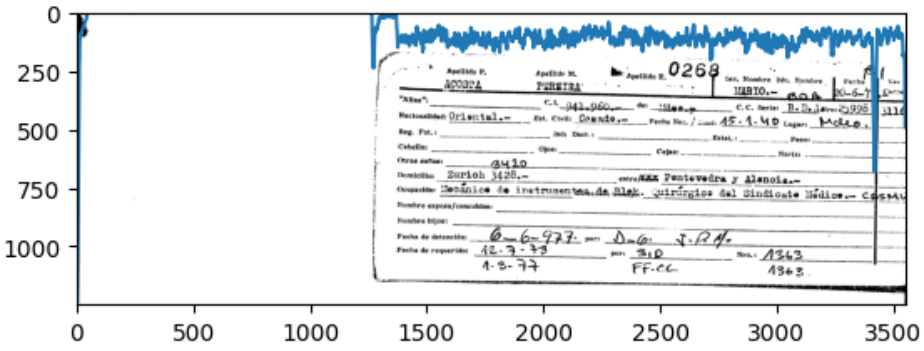



**Figura 4.21:** Primera parte de ficha de la hoja 287 del rollo 570, con el vector de suma de intensidades por columna.

En la Figura 4.22 se encuentra el mismo vector de intensidades por columna de la parte de ficha presente en la Figura 4.21 pero umbralizado. Se observan los últimos dos problemas mencionados: hay un pico en el borde aunque no hay parte de ficha y donde comienza la ficha aproximadamente en la posición 1300 hay un hueco.

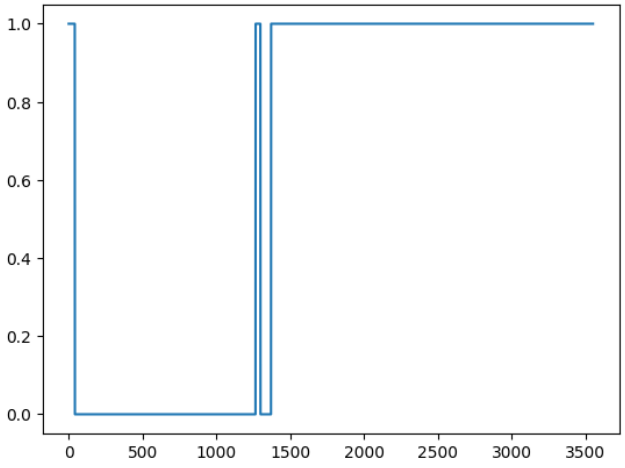



**Figura 4.22:** Vector C por columnas umbralizado de la primera parte de ficha de la hoja 287 del rollo 570.

En la Figura 4.23 se puede ver el resultado de aplicar los pasos para rellenar huecos y arreglar el ruido en los bordes. El resultado es un vector que contiene 1 donde hay parte de ficha y 0 donde no la hay. En este caso, la parte de ficha se encuentra sobre el lado derecho, desde un poco antes de la mitad de la hoja hasta el final.

En este momento, el vector solo debería valer 1 donde hay parte de ficha. Por lo

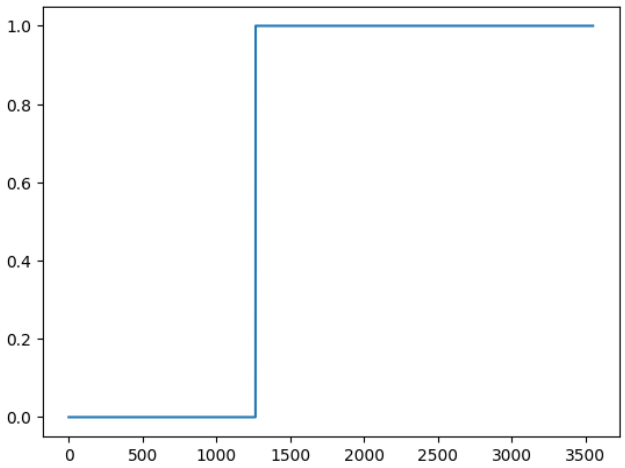



**Figura 4.23:** Vector C por columnas umbralizado y con huecos rellenados de la primera parte de ficha de la hoja 287 del rollo 570.

tanto, se busca el primer índice positivo y el último índice positivo. Finalmente, se recorta la parte de ficha verticalmente por esos índices.

### 4.1.7. Guardar cada parte de ficha en su propia imagen

Finalmente, se guarda cada parte de ficha en su propio archivo. Para nombrarlos, se utiliza el nombre de la imagen original y se le agrega el sufijo *_i* dónde *i* comienza en 0 y corresponde con el número de parte de ficha en la imagen.

Ejecutando en Cluster.uy, el proceso de extracción de partes lleva en promedio seis horas y media por rollo.

## 4.2. Clasificación de partes de ficha

Con las partes de ficha extraídas, se puede resolver el problema de clasificación y extracción de la información. Mediante observación de varias partes de ficha, se logra identificar 10 tipos diferentes de partes de adelante de ficha y 3 tipos diferentes de partes de atrás. Antes de extraer información, la parte de ficha es clasificada para poder conocer qué campos contiene y su ubicación.

Debido a que las imágenes no están etiquetadas se elige utilizar un algoritmo de *template matching* para clasificar las partes de ficha ya que no requiere etiquetar datos.

La clasificación de una parte de ficha se puede separar en dos problemas. El primero, dado un patrón y una parte de ficha, consiste en determinar si hay una ocurrencia del patrón y en qué posición ocurre. El segundo, una vez se tienen los coeficientes de similitud del conjunto de patrones de un tipo de ficha, determinar si esa parte de ficha es o no de esa clase.

Todas las partes de ficha de una misma clase tienen campos en común. Por ejemplo: nombre, apellido, fecha de nacimiento, etc. El objetivo es utilizar esos campos como patrones para clasificar la parte de ficha. Para cada clase de parte de ficha, se va a contar con un conjunto de patrones. Si para una nueva parte de ficha la cantidad de patrones con una coincidencia supera cierto umbral entonces podemos clasificar esa parte de ficha como de ese tipo.

En las Figuras 4.24, 4.25, 4.26, 4.27, 4.28, 4.29, 4.30, 4.31, 4.32 y 4.33 se encuentran las 10 clases de partes frontales de ficha encontradas. Se puede ver que las clases 1 y 3 son muy parecidas, cambia solamente la tipografía y un poco la posición de los campos. Las clases 2 y 8 no contienen los campos 'Fecha de detención' ni 'Fecha de requerido'. Se observa también que la mayor diferencia entre las clases se da en el orden de los campos o tipografía. A veces los espacios para completar los campos son una línea punteada (clase 4) y a veces una línea continua (clase 1).

Otra observación interesante es el campo Alias. El campo contiene comillas en todas las clases, sin embargo las comillas son diferentes. En ocasiones, se encuentra completamente en mayúsculas (clase 2 y clase 8).

Para generar los patrones, se busca una imagen con poco ruido de cada una de las clases y se recortan las etiquetas de los campos. Existieron algunas clases para las que fue necesario recurrir a una segunda imagen ya que no se encontró una imagen con poco ruido en todos los campos.

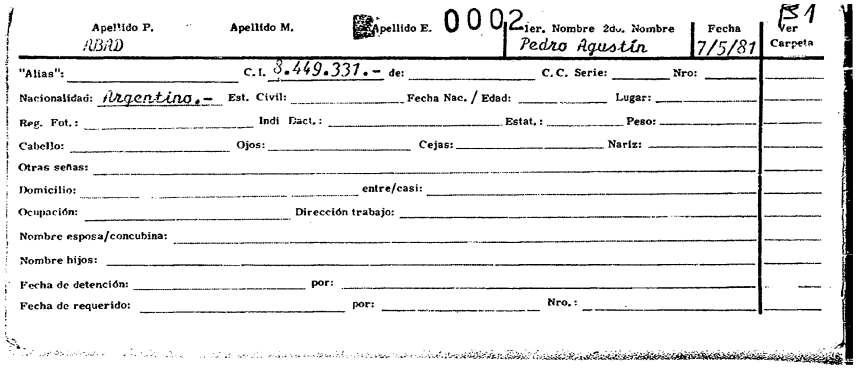



**Figura 4.24:** Ejemplo clase de ficha 1.

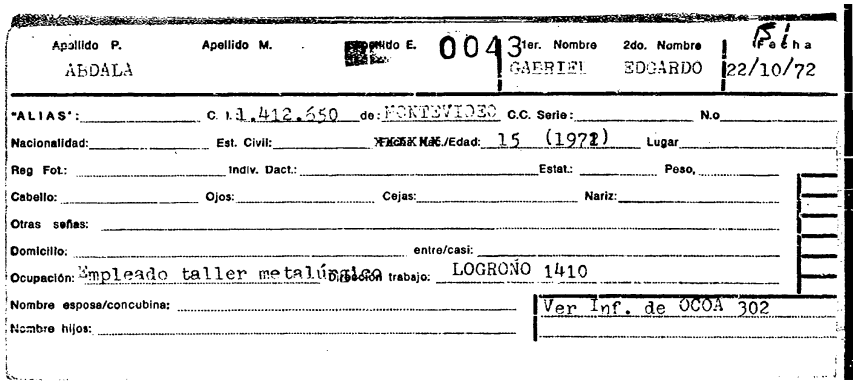



**Figura 4.25:** Ejemplo clase de ficha 2.

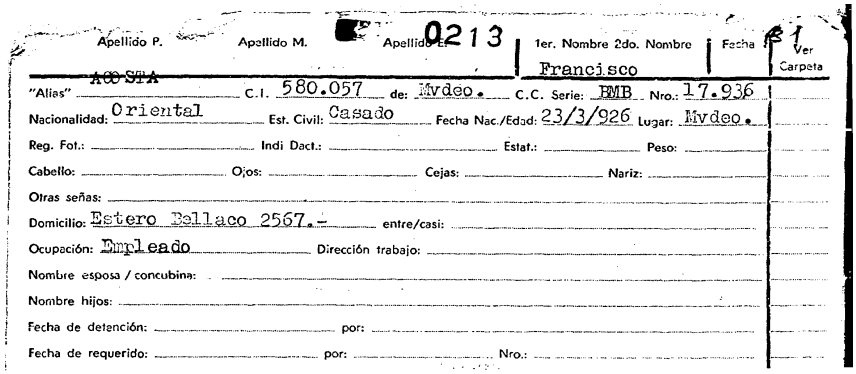



**Figura 4.26:** Ejemplo clase de ficha 3.

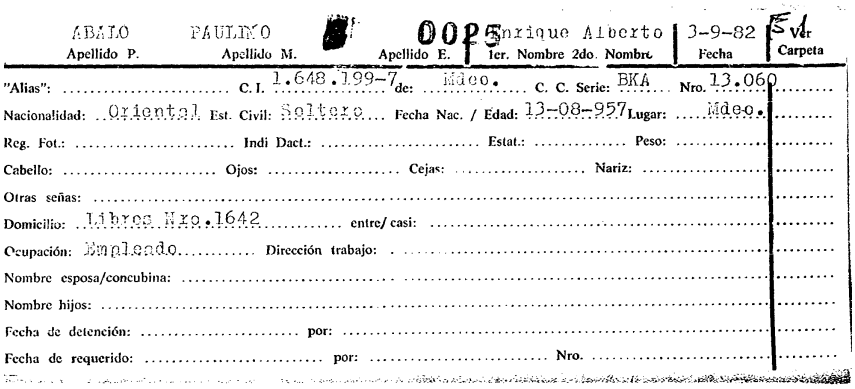



**Figura 4.27:** Ejemplo clase de ficha 4.

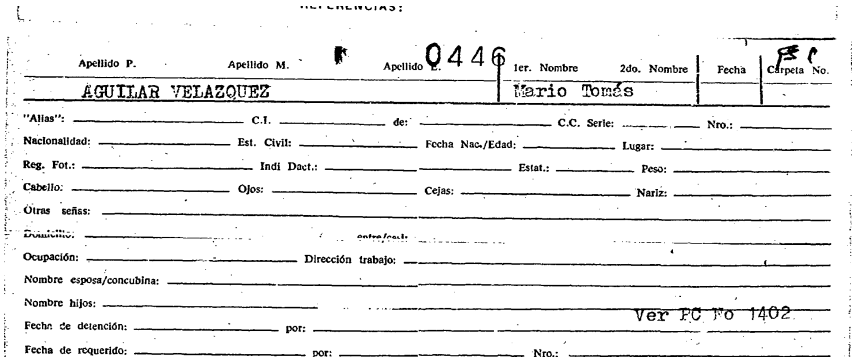



**Figura 4.28:** Ejemplo clase de ficha 5.

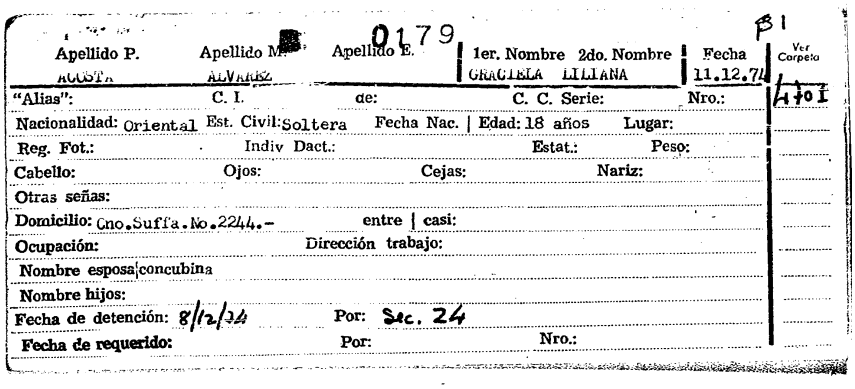



**Figura 4.29:** Ejemplo clase de ficha 6.

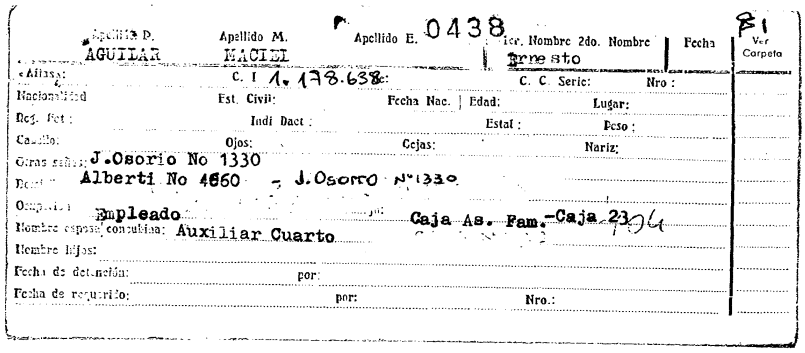



**Figura 4.30:** Ejemplo clase de ficha 7.

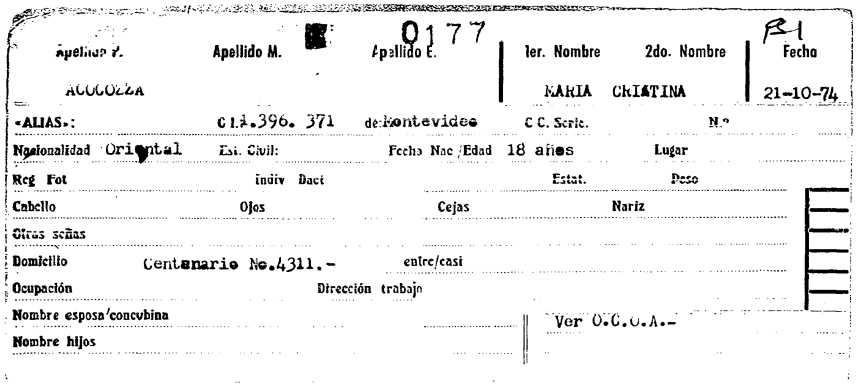



**Figura 4.31:** Ejemplo clase de ficha 8.

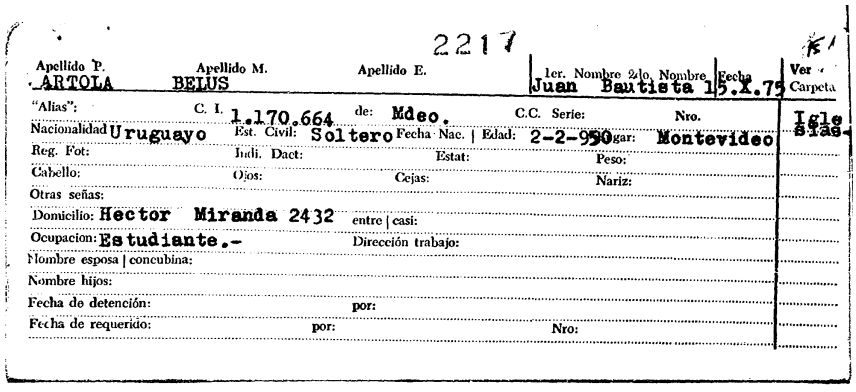



**Figura 4.32:** Ejemplo clase de ficha 9.

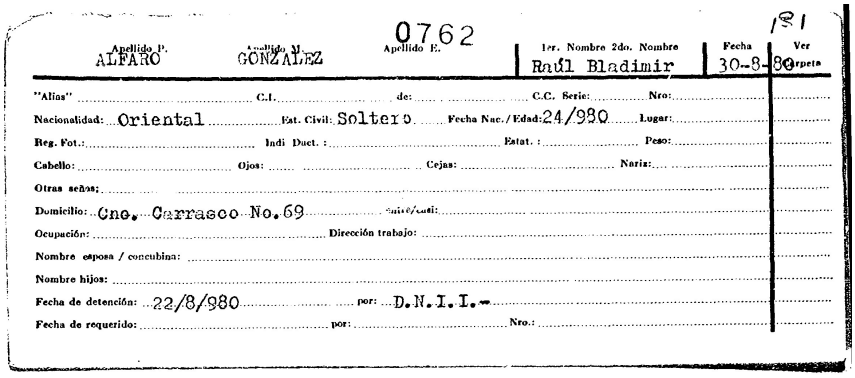



**Figura 4.33:** Ejemplo clase de ficha 10.

### 4.2.1. Generación de patrones

A continuación, se explica el proceso seguido para la generación de los patrones. Para cada clase (partes frontales y partes de atrás) se busca un ejemplo de cada clase que contenga poco ruido. Se busca utilizar como patrones el texto que describe los campos, es decir, las etiquetas de los campos. Además, se busca que cada patrón tenga el mayor tamaño posible. Es necesario realizar este paso con cada clase de parte de ficha. En la Figura 4.34 se encuentra una imagen que contiene todos los patrones seleccionados para la clase 1 de parte frontal.

En la Tabla 4.1 se muestra la cantidad de patrones que se utilizaron para cada clase. Se observa que se utilizó una cantidad similar para cada clase. Es importante destacar que cada patrón utilizado corresponde a un campo que se extrajo. En el caso de las partes de atrás, se agruparon todos los patrones ya que solo interesa detectar las partes de atrás sin determinar a cuál clase pertenecen.

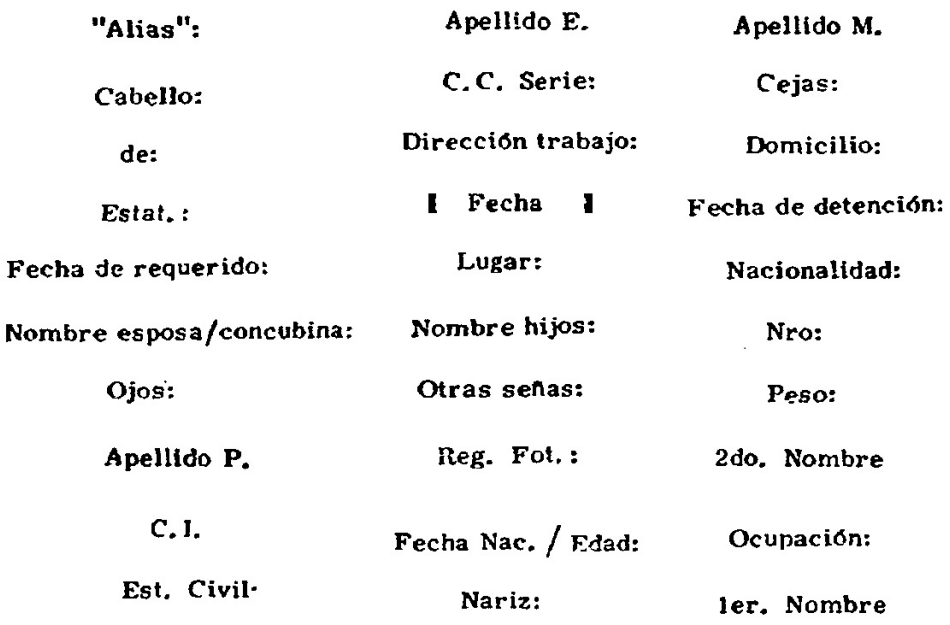



**Figura 4.34:** Patrones para la clase 1 de parte frontal de parte de ficha.

**Tabla 4.1:** Cantidad de patrones utilizados para cada clase.

| # Clase | Cantidad Patrones |
|---|---|
| Clase 1 | 30 |
| Clase 2 | 28 |
| Clase 3 | 32 |
| Clase 4 | 31 |
| Clase 5 | 31 |
| Clase 6 | 32 |
| Clase 7 | 33 |
| Clase 8 | 30 |
| Clase 9 | 33 |
| Clase 10 | 32 |
| Parte atrás | 10 |

### 4.2.2. Similitud entre imagen y patrón

Las medidas de similitud buscan determinar la ocurrencia o no del patrón en la imagen. Se consideraron dos medidas de similitud, distancia euclidiana y distancia coseno. Finalmente, se decidió utilizar distancia coseno debido a que mostró

mejores resultados en un pequeño experimento realizado. El mismo consistió en tomar cinco partes frontales de ficha, calcular los coeficientes de similitud utilizando ambas distancias. Se observó que en general la distancia coseno mostraba mayor similitud cuando se utilizaban los patrones de la clase correspondiente a esa parte de ficha.

En la Ecuación 4.2 se encuentra la fórmula para la distancia coseno entre dos vectores $A$ y $B$. El numerador contiene el producto interno entre los dos vectores y el denominador la multiplicación de la norma dos euclidiana de cada uno de los vectores. En el problema a resolver, el vector $B$ es un patrón y $A$ es un recorte de una parte de ficha del mismo tamaño que el patrón. Ambos son matrices de dos dimensiones.

$$cos\_sim(A, B) = \frac{A \cdot B}{\|A\|\|B\|} \tag{4.2}$$

Dado que no conocemos la posición donde se encuentra el patrón, se calcula la similitud con todos los posibles recortes de imagen del mismo tamaño que el patrón. Es decir, se calcula la similitud del patrón con todas las posiciones posibles en la imagen. Una forma de realizar este cálculo es utilizando bucles. Sin embargo, esta solución es demasiado costosa computacionalmente, por lo que se busca utilizar una alternativa más eficiente. Una alternativa, es considerar al patrón y la imagen como señales. De esta manera, el producto interno del patrón con todos los recortes de la imagen del tamaño del patrón es el cálculo de la correlación. Gracias al teorema de convolución (Burger y Burge, 2016), la correlación se puede calcular de manera eficiente. Este teorema dice que una convolución lineal en el espacio de imágenes es equivalente a una multiplicación punto a punto en el espacio de frecuencia. Por lo tanto, hay que aplicar la transformada de Fourier para pasar al espacio de frecuencia, realizar la multiplicación punto a punto y aplicar la transformada de Fourier inversa para volver a la imagen.

Volviendo al problema, hay que calcular la correlación, no la convolución. Sin embargo, la correlación lineal es equivalente a la convolución lineal pero invirtiendo el filtro (Burger y Burge, 2016). En este caso, el filtro es la imagen del patrón. En la implementación se utiliza la biblioteca scipy («SciPy», s.f.) de Python ya que

contiene una función para calcular de manera eficiente la correlación aprovechando las propiedades mencionadas y utilizando el algoritmo de transformada rápida de Fourier.

El resultado de la correlación entre el patrón y la imagen entera es una matriz que contiene el cálculo del producto interno entre el patrón y la imagen colocando el patrón en todas las posiciones posibles. Si esta matriz se divide entre la norma del patrón y la norma de la subimagen (la imagen restante de considerar solo los píxeles de la posición donde se está colocando el patrón en el momento) entonces se obtiene una matriz de similitud $G$, del mismo tamaño que la imagen que en la posición $(i, j)$ va a contener el valor de similitud de la imagen con el patrón si se colocara el centro del patrón en ese punto.

Dado que la norma del patrón es la misma en todos las posiciones, se puede calcular y dividir al patrón antes de calcular la correlación con toda la imagen.

La norma de la subimagen no es tan directa de calcular porque es diferente para cada posición en la que se coloca el patrón. Una forma de resolver el problema es considerar una señal auxiliar del mismo tamaño que el patrón que contiene $1$ en todas las posiciones. El producto interno de esta señal con el cuadrado de una subimagen da como resultado la norma al cuadrado de la subimagen. Si se calcula este producto interno en todas las posiciones posibles entonces se va a obtener la norma al cuadrado de la subimagen en todas las posiciones posibles. Este cálculo es la correlación entre el cuadrado de la imagen y la señal auxiliar. Finalmente, se aplica raíz cuadrada al resultado para obtener la norma.

La formula final para calcular la matriz $G$ para una imagen y un patrón se encuentra en la Ecuación 4.3. $correlate$ representa la correlación. La variable $template1$ representa la señal auxiliar, es una matriz del mismo tamaño que el patrón que contiene $1$ en todas las posiciones. Se toma el máximo entre $1$ y la correlación para evitar dividir entre cero. Esto no afecta el resultado, ya que si la norma de la subimagen es $0$ entonces la subimagen contiene solo $0$ y por lo tanto el producto interno con el patrón (el numerador) también va a ser 0. Una vez calculada la matriz $G$ con la similitud para todos los posibles lugares dónde se puede colocar el patrón, se busca la posición donde la similitud es más alta. En el caso de la distancia coseno, esto es el máximo.

$$G = correlate(imagen, \frac{patron}{\|patron\|}) / \sqrt{max(1, correlate(imagen^2, template1))} \tag{4.3}$$

Para mejorar los resultados, se aplica un filtro pasa bajos gaussiano (desenfoque) tanto al patrón como a la imagen (parte de ficha), con un valor de $\sigma$ igual a $2$. Este filtrado contribuye a reducir el ruido en las imágenes permitiendo que el cálculo de la similitud se centre en los detalles más gruesos, como la estructura, en lugar de los pequeños detalles que podrían considerarse ruido. Se observó que valores más pequeños de $\sigma$ no alteraban mucho el resultado de la correlación mientras que utilizar valores más altos desenfocaban mucho la imagen perdiendo mucho nivel de detalle. Además, se rota el patrón de $-1$ a $1$ grados con un paso de $0,5$ y se calcula la matriz de similitud para todas las rotaciones. Para cada posición de la matriz de similitud, se utiliza el máximo de todas las rotaciones.

Para acelerar el algoritmo, se utiliza un enfoque multiescala. Esto consiste en hacer una búsqueda en una escala más chica con pasos más gruesos y después refinar en torno al lugar encontrado en la escala previa. Al realizar la búsqueda en una escala más pequeña el tiempo de ejecución se reduce. Primero se toma la imagen a escala $0,95$ y patrones a escala $0,95$, con rotaciones con un paso de $0,5$. Al encontrar el lugar donde se da la mayor similitud, se toma esa sección junto con un margen y se busca de nuevo el lugar donde se da la máxima similitud pero utilizando la imagen y patrones a tamaño completo y con rotaciones cada $0,1$ grados en esa zona.

En caso de que el coeficiente de similitud máximo supere un determinado umbral, entonces se dice que hay un *match* o una coincidencia para ese patrón en esa parte de ficha y en la posición donde se da el máximo. Para partes frontales de ficha, el umbral que la similitud coseno de un patrón debe superar para que haya una coincidencia es $0,93$ y en el caso de las partes de atrás el umbral utilizado es de $0,92$. Estos valores se determinaron mediante experimentación. Se calculó el coeficiente de similitud de los patrones de una clase para 100 ejemplos positivos y 100 ejemplos negativos. Luego, se realizó un gráfico del histograma del coeficiente de similitud para cada uno de los patrones. Se observa que $0,93$ logra separar los ejemplos positivos de los negativos para la mayoría de los patrones.

### 4.2.3. Identificación de parte de ficha a partir de similitudes

Dados los coeficientes de similitud de todos los patrones asociados a una clase con una parte de ficha, es necesario definir cuándo se puede clasificar esa parte de ficha como de esa clase.

En caso de ser patrones asociados a una clase de parte frontal, si hay un mínimo de 13 patrones que superan el umbral, entonces esa parte de ficha se clasifica como una parte frontal de esa clase. Este número se obtuvo nuevamente mediante experimentación. Se tomaron cinco partes de ficha de cada clase y se realizó la clasificación probando varios umbrales. El valor con el que se obtuvieron mejores resultados fue 13. Un valor mayor aumentaba la cantidad de falsos negativos y un valor menor aumentaba la cantidad de falsos positivos.

Para partes de atrás, alcanza con que un solo patrón supere el umbral para clasificar una parte de ficha como parte de atrás. Se toma esta decisión dado que los patrones de la parte de atrás son muy característicos y es poco probable que se detecten en la parte frontal.

A la cantidad de patrones con coincidencia necesarios para clasificar la imagen se le llama umbral de clasificación.

En la Figura 4.35 se encuentra un ejemplo de una parte de ficha junto con los patrones detectados. La parte de ficha se encuentra sombreada excepto en donde se detectaron patrones.

### 4.2.4. Algoritmo de clasificación

De lo explicado en las secciones 4.2.2 y 4.2.3 surge el algoritmo de clasificación utilizado. Dada una parte de ficha a clasificar, para cada clase existente (es decir, que se cuenta con un conjunto de patrones asociados a ella) se calculan los coeficientes de similitud para cada uno de los patrones. Luego, para cada clase se verifica cuántos patrones superan el umbral de similitud y si la suma de la cantidad de esos patrones supera el umbral de clasificación.

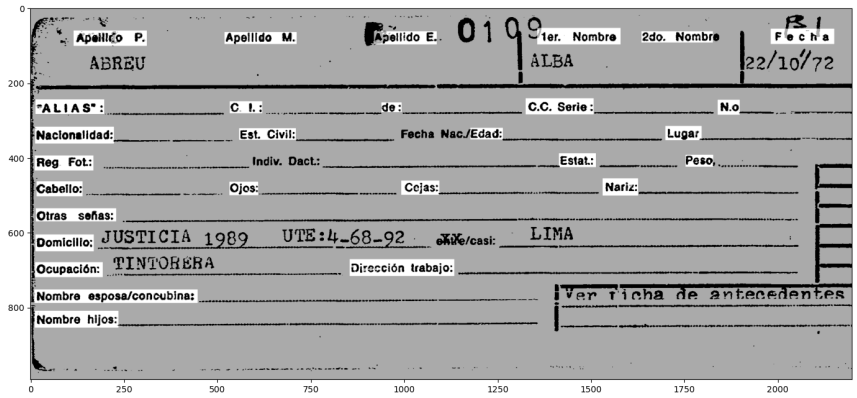



**Figura 4.35:** Parte de ficha 1 de la imagen 130 del rollo 570 junto con las ubicaciones de los patrones detectados.

Cuántos menos patrones se prueben, más rápida va a ser la clasificación. Por lo tanto, se busca una estrategia que verifique la menor cantidad de patrones. Ya que la mayoría de las fichas tienen una parte frontal y una parte de atrás, se puede asumir que, aproximadamente, la mitad de las partes de ficha van a ser partes de atrás. Además, 'parte de atrás' es la clase que tiene menos patrones, por lo que al momento de clasificar una imagen, primero se verifica si es una parte de atrás y en caso de que no lo sea, se continúa verificando si es alguna de las partes frontales conocidas.

Para las clases asociadas a parte frontal, los patrones se prueban según la frecuencia de la clase, utilizando primero los pertenecientes a las clases más frecuentes. En caso de que una imagen sea clasificada en una clase, no se prueba con los patrones de las restantes clases. Existe también la posibilidad de que una imagen no se clasifique en ninguna clase. Para estos casos se tiene una categoría 'otros'.

Utilizando esta estrategia, la clasificación va a ser más rápida para partes de atrás y va a ser más lenta si la parte de ficha corresponde a las clases con menos ocurrencias.

En el listado 4.1 se encuentra el pseudocódigo de este algoritmo.

```
UMBRAL_MATCH = x;
UMBRAL_CLASIFICACION = y;
UMBRAL_MATCH_PARTE_ATRAS = z;

patrones_1 = leer_patrones("clase1");
...
...
patrones_10 = leer_patrones("clase10");

patrones_partes_atras = leer_patrones("partes_atras");
patrones_clases = [patrones_1, ...., patrones_10];


algoritmo_clasificacion(imagen_candidata) {
  for imagen_patron in patrones_partes_atras {
    if (coeficiente_similitud(imagen_patron, imagen_candidata) >= UMBRAL_MATCH_PARTE_ATRAS) {
      return "parte_atras";
    }
  }

  for (nombre_clase, patrones) in patrones_clases {
    patrones_matcheados = 0;
    for imagen_patron in patrones {
      if (coeficiente_similitud(imagen_patron, imagen_candidata) >= UMBRAL_MATCH) {
        patrones_matcheados += 1;
      }
    }

    if (patrones_matcheados >= UMBRAL_CLASIFICACION) {
      return nombre_clase;
    }
  }

  return null;
}
```

**Listing 4.1:** Pseudocódigo algoritmo de clasificación

### Detección de colisiones

Debido a ruido en las partes de ficha, es posible que haya dos patrones que obtengan una coincidencia en posiciones cercanas, haciendo que los patrones se superpongan. En estos casos, se dice que hay una colisión de los dos patrones. Si esto sucede, sabemos que hay uno de los patrones que quedó mal posicionado. Para solucionarlo, se deja en ese lugar el patrón con una similitud más alta y para el otro patrón se busca el máximo ignorando esa zona. En la implementación, luego de encontrar la posición de máxima similitud para cada patrón, se toman todas las combinaciones de a dos patrones posibles y se revisa en cuales hay colisiones. En caso de existir, se mira cuál de los dos patrones tiene una similitud más baja y se busca el segundo máximo para ese patrón. Se utiliza nuevamente la matriz de similitud, buscando el máximo luego de poner 0 donde se encuentra la coincidencia del patrón con mayor similitud. El proceso se realiza en dos etapas, primero se

genera el conjunto de patrones a los que hay que buscar nuevo máximo y luego se realiza la búsqueda. Este proceso se repite hasta que no haya más colisiones o se alcanza un máximo de 15 iteraciones, siendo cada iteración una comprobación de todas las combinaciones y búsqueda del nuevo máximo restringiendo el área de búsqueda.

#### Optimizaciones

Con el objetivo de mejorar la velocidad de procesamiento, se realizan algunas optimizaciones.

Una primer optimización implementada es evitar buscar coincidencias de patrones innecesariamente. Al calcular los coeficientes de correlación para los patrones de una clase, si en algún momento la cantidad de patrones con coincidencia sumada con la cantidad de patrones restantes a buscar es inferior al umbral, entonces se continúa con los patrones de la siguiente clase ya que no va a ser posible alcanzar el umbral.

La segunda optimización realizada surge de observar que la tarea de clasificación es altamente paralelizable. En particular, se optó por clasificar en simultáneo varias partes de ficha, utilizando un núcleo de CPU para cada imagen. Para obtener más capacidad de cómputo se utilizó Cluster.uy que contiene servidores con hasta 40 núcleos.

### 4.2.5. Extracción de campos

Una vez clasificadas las partes de ficha, se pueden extraer los campos. Como resultado del *template matching*, se conoce la ubicación de los patrones. Debido a que los patrones que se utilizaron corresponden a las etiquetas de los campos a extraer y los mismos están en una posición definida respecto a ellos, ver Figura 4.35, entonces se conoce para cada ficha clasificada la ubicación de los campos.

Es necesario definir para cada clase de parte de ficha y para cada patrón (es

decir, cada campo), qué largo tiene el espacio en blanco y a qué lado del patrón se encuentra el espacio en blanco donde se encuentra la información a extraer.

Debido a que existen casos en los que el texto no se encuentra alineado a la descripción del campo (es decir al patrón detectado) y existen casos en los que no se respetan los renglones de la plantilla, es necesario aproximar la ubicación de la línea de texto. Nuevamente se hace uso de la suma de intensidades por fila para abordar este problema.

Dado un patrón ubicado asociado a un campo, se considera el rectángulo donde se encuentra el texto asociado a ese campo. Se utiliza una altura de 100 píxeles, que es mayor al tamaño de una línea. El objetivo es que si el texto se encuentra por arriba o por abajo del renglón de todas maneras quede en el rectángulo considerado.

Sobre ese rectángulo se calcula la suma de intensidades por fila y se realiza una umbralización con un umbral de $5$. El objetivo es obtener $1$ en donde hay texto y $0$ donde no hay texto, obteniendo como resultado una gráfica de pulsos.

Luego, se buscan los índices donde ocurre el pulso más ancho y se recorta la imagen sobre esos índices, agregando un pequeño margen. En caso de que el pulso más ancho sea menor a 50 píxeles, se recorta desde los bordes del rectángulo considerado hasta obtener una imagen con la misma altura que un renglón.

Esta técnica se basa en la observación de que si el texto se encuentra desalineado con el renglón, entonces el pulso generado por el texto va a ser más ancho que el generado por la línea del renglón. Además, si el recorte considerado contiene una parte de la línea de texto superior o inferior, como no son completas, el pulso generado por ellas va a ser de menor tamaño que el pulso de la línea de texto que se busca extraer.

Con esta información, se pueden extraer los campos. Cada uno de ellos se almacena en una imagen por separado. Esta etapa se realiza inmediatamente después de clasificar la parte de ficha.

En la Figura 4.36 se encuentra nuevamente la ficha 1 de la imagen 130 del rollo 570, con los patrones detectados y los campos que se extraen marcados. La

mayoría de los espacios a completar de los campos se encuentran a la derecha de la descripción del mismo. Por ejemplo, los campos nacionalidad, cédula, cabello. También se observan campos en que el espacio en blanco se encuentra abajo de la etiqueta, como es el caso de los apellidos y nombres. No ocurre en este caso, pero también hay clases de partes frontales dónde el espacio para completar algunos campos se encuentra arriba de la etiqueta.

En el caso de los campos extraídos correspondientes al primer y segundo nombre, se decide extraer ambos campos juntos. De esta manera, se obtiene redundancia. Si alguno de los dos patrones no se encuentra por ejemplo, debido a ruido, de todas maneras se extraen los dos nombres. Además, sucede que el límite de donde termina el primer nombre y comienza el segundo depende de cómo fue completada la ficha y por lo tanto tampoco es uniforme en todas las fichas, lo que puede provocar extraer nombres cortados si se busca extraerlos por separado.

Con los campos 'Apellido Paterno', 'Apellido Materno' y 'Apellido E.' se aplica una estrategia similar, de manera de obtener redundancia para los apellidos y de evitar extraer apellidos cortados.

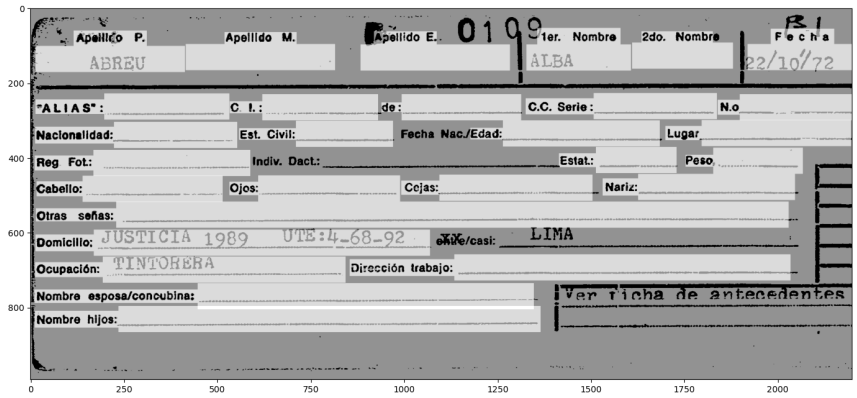



**Figura 4.36:** Parte de ficha 1 de la imagen 130 del rollo 570 junto con las ubicaciones de los patrones detectados y los campos a extraer marcados.

Este proceso se ejecutó en Cluster.uy. En promedio, cada rollo tomó cinco horas en procesarse, ejecutando la clasificación en paralelo con 36 núcleos (es decir, se

realizaba la clasificación de a 36 imágenes simultáneamente). Esto da un tiempo promedio de 153 segundos para tratar una parte de ficha.

## 4.3. Extracción de número de ficha

Como se mencionó previamente, cada parte de ficha, tanto si es parte frontal como parte de atrás, contiene un número de cuatro o cinco dígitos que identifican todas las partes de ficha asociadas a una misma persona. Además, dicho número sigue un orden secuencial y alfabético, comenzando en 0001 y en el índice de cada rollo mediante el uso de ese número se puede obtener información adicional sobre la persona, como su nombre y organización. Este último dato no se encuentra disponible en la ficha. Es por este motivo que resulta de particular importancia lograr extraer este número y reconocer el texto.

Sin embargo, esta tarea se ve dificultada por varios motivos. Dado que el número no se encuentra al lado de un campo como sucede con todos los demás datos que se extraen, no es posible tratar este dato como el resto de la información y extraerlo mediante la ubicación de un solo patrón en la ficha. Otro problema radica en que la posición es variable, lo que impide localizar un vértice de la parte de ficha y calcular su ubicación en base a él. Por último, es posible que el número se encuentre ligeramente rotado en relación al resto del texto, dificultando su detección.

Afortunadamente, se pueden aprovechar algunas características importantes de estos números. Como consecuencia de encontrarse ordenados, es posible determinar fácilmente a partir de qué imagen los números pasan de tener cuatro a tener cinco dígitos. Esto implica que, al tener una parte de ficha específica, se puede determinar la cantidad de dígitos que contiene su número en función del rollo, la hoja y la posición en la hoja en la cual se encontraba al momento de la separación. La segunda característica relevante, es que los dígitos poseen todos una tipografía muy similar, lo que permite utilizar un único patrón para cada dígito a la hora de realizar *template matching*. Finalmente, los dígitos tienen todos el mismo tamaño en todas las partes de ficha.

Mediante experimentación, se descubre que al realizar *template matching* con

los diez patrones asociados a los dígitos del número sobre el tercio superior de la imagen, en la mayoría de los casos el patrón con mayor similitud es correctamente posicionado sobre uno de los dígitos. Es posible que el dígito detectado no corresponda al patrón, no obstante de todas formas se obtiene la ubicación de un dígito. Una vez conocida la posición del primer dígito, se puede reducir la búsqueda a un área acotada equivalente a tres o cuatro dígitos a la derecha e izquierda de donde se detectó el primer dígito.

No se tiene en cuenta el patrón asociado al número 1 al detectar el primer dígito, debido a que en muchos casos tiene una similitud muy alta con la plantilla. En partes de ficha cuyo número contiene 1 a veces incluso es mayor la similitud con una línea recta de la plantilla que con el dígito.

Se observa también que el número en general se encuentra sobre los bordes izquierdos o derechos de la imagen, por lo que además de considerar solo el tercio superior, se recortan algunos píxeles del lado izquierdo y derecho.

En la Figura 4.37 se encuentra para la primera parte de ficha de la hoja 119 del rollo 570 el resultado de aplicar *template matching* en esta zona reducida. El número de ficha es 0096. Se puede ver que los patrones del cero, el nueve y el seis fueron posicionados en el lugar correcto. El patrón del ocho obtiene su similitud máxima en la posición del primer cero. Además, se observa que el patrón del cuatro, uno y el siete obtienen su similitud máxima al colocarse por encima de parte de la plantilla de la ficha.

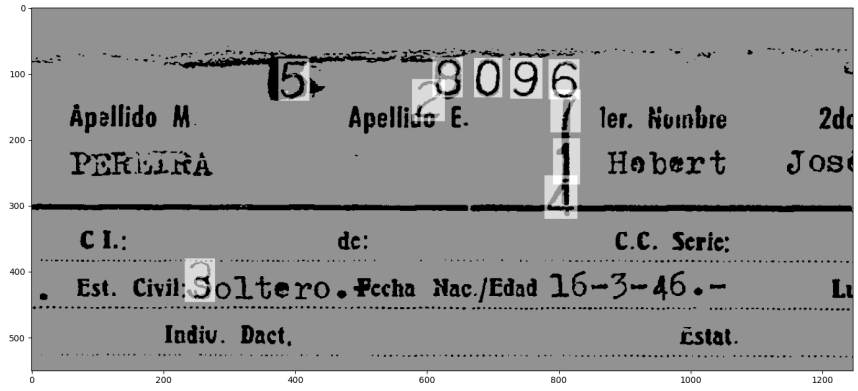



**Figura 4.37:** Resultado de ejecutar *template matching* con el objetivo de detectar el número de ficha para la primera parte de ficha de la hoja 119 del rollo 570.

Sin considerar el patrón del uno, el que obtuvo una similitud más alta fue el del nueve.

A continuación, se explican los dos métodos utilizados y sus diferencias.

### 4.3.1. Extracción mediante template matching múltiple y umbral

Este método consiste en aplicar nuevamente *template matching* sobre un área reducida de la imagen. Una vez obtenido el primer dígito mediante *template matching* en la parte superior, se buscan dígitos hacia la izquierda y derecha iterativamente.

En primer lugar, se considera un rectángulo hacia la derecha del dígito detectado, con una altura ligeramente mayor que la de un dígito y del ancho de un dígito. Luego, se aplica *template matching* con los diez patrones en ese rectángulo. Si se supera un umbral determinado, se busca en otro rectángulo a la derecha del segundo dígito. Este proceso finaliza cuando no se supera el umbral o se alcanza la de dígitos en parte de ficha (dato conocido).

A continuación, se realiza el proceso análogo pero hacia la izquierda. Es importante notar que no se buscan dígitos hasta completar los que no se encontraron hacia la derecha, sino que se busca hacia la izquierda nuevamente hasta superar el umbral o encontrar la cantidad de dígitos en la parte de ficha.

En la Figura 4.38 se puede observar sombreada la zona de búsqueda de dígitos explicada anteriormente para la primera parte de ficha de la hoja 119 del rollo 570.

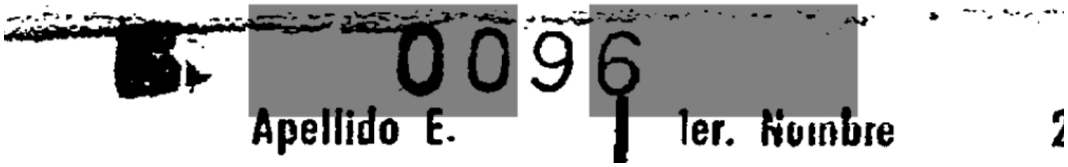



**Figura 4.38:** Zona de búsqueda de dígitos para la primera parte de ficha de la hoja 119 del rollo 570.

La imagen resultante con el número extraído con este método para la primera parte de ficha de la hoja 119 del rollo 570 está disponible en la Figura 4.39.

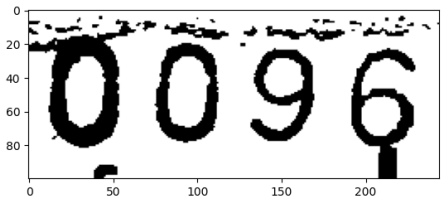



**Figura 4.39:** Número extraído de la primera parte de ficha de la hoja 119 del rollo 570 utilizando el método de *template matching* múltiple y umbral.

Es posible que se detecten más dígitos de los que contiene la parte de ficha al juntar la búsqueda hacia la izquierda y hacia la derecha. Sin embargo, en caso de encontrar por ejemplo, tres dígitos hacia cada lado, no es fácil decidir cuáles descartar, ya que todos superaron el umbral establecido. Aumentar el umbral puede empeorar la detección debido a la gran cantidad de ruido del problema.

Otra situación posible es que se tenga por ejemplo, una parte de ficha en el que un dígito no tiene un coeficiente de similitud alto con ninguno de los patrones, lo cual evita que se sigan buscando dígitos a la derecha o izquierda de ese rectángulo, obteniendo como resultado menos dígitos.

### 4.3.2. Extracción mediante template matching múltiple y maximización en ventana

Para el segundo método, se consideran rectángulos a la derecha y a la izquierda del dígito detectado. Estos rectángulos tienen el ancho de un dígito y una altura ligeramente mayor que la altura de un dígito. La cantidad de rectángulos a cada lado es igual a todos los posibles dígitos que puede haber a la izquierda o derecha del primer dígito, es decir, tres o cuatro según la imagen. El resultado es un arreglo de imágenes de siete u ocho dígitos. En este momento, se realiza *template matching* en cada rectángulo, considerando en esta ocasión el patrón asociado al uno, y se genera un arreglo en el que cada posición contiene la máxima similitud detectada en cada rectángulo.

A continuación, se busca maximizar, en el vector, la suma de los valores de la correlación dentro de una ventana cuyo tamaño es igual a la cantidad de dígitos en la imagen. Finalmente, se recorta la imagen original donde comienza el dígito de la ventana máxima de más a la izquierda hasta el final del dígito de la ventana máxima situado más a la derecha.

Por ejemplo, para la primera parte de ficha de la hoja 119 del rollo 570, el vector resultante es (0.57, 0.60, 0.85, 0.95, 0.97, 0.93, 0.45, 0.44, 0.47). La imagen tiene cuatro dígitos. Por lo tanto, el máximo es 3,7 y se alcanza al sumar desde la posición que contiene el valor 0,85 hasta la posición con el valor 0,93.

La imagen resultante con el número extraído de esta parte de ficha utilizando el método de *template matching* múltiple y maximización en ventana se encuentra en la Figura 4.40.

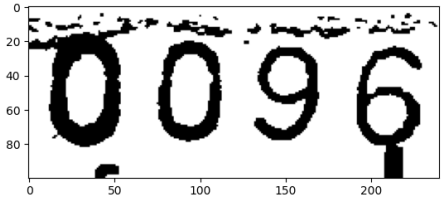



**Figura 4.40:** Número extraído de la primera parte de ficha de la hoja 119 del rollo 570 utilizando el método de *template matching* y maximización en ventana.

Algunas observaciones con respecto al método de la sección 4.3.1. En este caso, el resultado va a tener el largo necesario para la cantidad de dígitos en la parte de ficha. Nunca habrá espacio sobrante que genere ruido más tarde al intentar reconocer los dígitos. Otra diferencia es que no es necesario establecer un umbral, más allá del utilizado para detectar el primer dígito. Como consecuencia, es posible que un dígito tenga un coeficiente de similitud bajo y, sin embargo quede dentro de la ventana máxima gracias a la similitud de sus dígitos vecinos.

Se utilizaron ambas técnicas en todas las partes de ficha extraídas. Resulta importante volver a resaltar que si bien se detectaron los dígitos mediante *template matching*, el resultado final de esta etapa es para cada parte de ficha una imagen que contiene el número. A esta imagen luego se le aplica OCR. No se utilizó el resultado del *template matching* como los dígitos resultantes debido a que el OCR mostró un mejor desempeño. Por la presencia de ruido, en ocasiones un dígito incorrecto obtenía mayor similitud que el correcto. También es importante resaltar que ninguno de los dos métodos funciona en caso de que el número no se encuentre en el tercio superior de la parte de ficha, aunque esta situación es poco frecuente.

El tiempo requerido para extraer los números de todos los rollos y usando ambos métodos fue de ocho horas. Se ejecutó en paralelo la extracción en 36 núcleos utilizando los servidores de Cluster.uy.

## 4.4. Procesamiento del índice

En esta sección se detalla el trabajo realizado con el fin de extraer información del índice. Cómo se mencionó anteriormente en el capítulo 3, cada rollo incluye un índice que contiene número de ficha, nombre de la persona, organizaciones a las que pertenecía, observaciones y número de carpeta. Esto resulta de relevancia porque las organizaciones a las que pertenecía la persona se encuentran en el índice y no en la ficha. Además, la tipografía es uniforme en todas las hojas de los índices, algo que no pasa en las fichas, por lo que si se logra reconocer el texto del índice y utilizar el número de ficha para asociar cada línea con la correspondiente ficha se logra obtener redundancia de datos en cuanto al nombre de la persona.

El índice está compuesto de hojas impresas en formato tabla. Cada columna tiene cierto largo fijo. Por ende, si se detecta el lugar de comienzo del texto en la hoja junto con los espacios entre las líneas se puede separar el índice en filas y columnas para aplicarle un OCR.

Primero mediante observación se define para cada rollo un margen sobre el lado izquierdo de la hoja. El objetivo de este margen es reducir posible ruido y se busca que se encuentre lo más cerca del comienzo del texto posible. El margen es diferente para cada rollo debido a que al momento de adquirir las imágenes en algunos rollos se optó por dejar más espacio en blanco a la izquierda. El margen más pequeño es de 0 píxeles y el más grande de 530.

En este momento, se aplica el algoritmo de enderezado utilizado para las partes de ficha de la sección 4.1.4 para toda la hoja, de manera de dejar las líneas lo más derechas posibles.

Además, hay algunas hojas que se encuentran rotadas 90º. Este problema se soluciona fácilmente mirando el largo y ancho de la imagen antes de ejecutar el algoritmo de enderezado.

Luego de este preprocesamiento, se pasa primero a hallar el índice donde comienza el texto y luego la separación en líneas de texto.

Para el primer problema, se utiliza un algoritmo similar al generado para separar las partes de ficha de una hoja. Se define un margen superior e inferior. Además de un margen del lado derecho. Estos márgenes son los mismos para todos los rollos.

Luego, se calcula la suma de intensidades por columnas y se busca maximizar una ventana de negro del mismo largo que una línea del texto. Se considera toda la altura restante de la hoja luego de eliminar los márgenes.

El índice donde comienza esta ventana máxima debe coincidir con la posición donde comienza el texto. No obstante, debido a ruido es posible que el valor máximo se de en varias ventanas diferentes. En caso de que esto suceda, se observó que tomar la ventana que se encuentra más hacia la derecha daba mejores resultados.

Ahora se recorta la hoja enderezada desde índice hasta el final, sumándole el largo de una línea y se procede a separar en renglones. Se comienza por tomar solamente la columna que contiene los números, esto se hace para disminuir el ruido. Para encontrar la separación de líneas, se busca hallar los índices de los espacios en blanco que separan las líneas. Esto es más robusto que detectar los renglones ya que el texto puede no encontrarse totalmente alineado con los renglones e incluso en algunas hojas del índice los renglones están borrados. Sin embargo, siempre va a haber espacio en blanco entre dos líneas. Esto permite utilizar el algoritmo de segmentación de líneas mencionado en la sección 2.1.1.

Se realiza el cálculo de la suma de intensidades por filas y se buscan máximos en el vector resultante (en este caso, el valor blanco se representa como $1$ y el negro como $0$). Algunos de estos puntos donde se encuentran los máximos significan la separación entre dos líneas. Para eliminar los puntos que no representan separación, se busca que la distancia entre dos máximos sea de por lo menos $35$, que es aproximadamente la altura de una línea. Si hay dos máximos a una distancia menor a ese valor, se considera solo el máximo más alto. Finalmente, se divide la hoja en líneas y cada línea se divide en columnas, las cuales se guardan en archivos de imagen individuales.

A cada una de estas imágenes se le aplica un OCR para convertir su contenido a texto.

El tiempo total necesario para guardar cada columna en su imagen individual fue de dos horas y media. Este proceso se ejecutó utilizando un núcleo solo de CPU y se realizó en un equipo con un procesador i9 de onceava generación y 16 GB de ram.

## 4.5. Reconocimiento de texto

El paso final consiste en procesar tanto los campos extraídos de las partes de ficha como el índice de cada rollo recortado para poder convertir su contenido a texto.

En el caso de las partes de ficha, se hace foco en obtener buenos resultados para los campos asociados al nombre y a la cédula, junto con el número de ficha extraído. Estos campos contienen información que permite identificar a quien pertenece la ficha. En el caso del índice, se busca lograr buenos resultados con la columna número y la columna nombre.

Para intentar reconocer cédulas se prueba utilizar reconocimiento de patrones y OCR. Para el resto de las imágenes solo se utiliza OCR.

### 4.5.1. template matching para reconocimiento de cédulas

Para obtener los dígitos de las cédulas se intenta realizar un *template matching*, similar a como se hizo para la clasificación de partes de ficha y extracción de campos. Recordando de la sección 2.1, lo que se realiza es un OCR con segmentación de caracteres. Se buscan patrones dentro de las imágenes de las cédulas con el objetivo de detectar los dígitos y en el orden correcto.

Los patrones que se utilizan son dígitos, puntos y guiones recortados de imágenes del campo cédula de identidad. Hay varias tipografías y no siempre los dígitos de una tipografía generan una buena coincidencia con una tipografía distinta. Además, es posible que haya ruido en las imágenes que dificultan la detección. Por lo

tanto, se utilizan mucho más que 12 patrones (en el caso que todas las cédulas fueran de la misma tipografía y no hubiera ruido, existiría un patrón por dígito, uno para punto y uno para guion). En total, se generaron 152 patrones.

En la Figura 4.41, se encuentra el campo cédula de una parte de ficha. En este caso, si se generaran patrones con esta imagen se obtendrían 7: un patrón para el dígito 8, uno para el dígito 4, uno para el dígito 9, uno para el dígito 3, uno para el dígito 1, un patrón para el punto y finalmente un patrón para el guion.

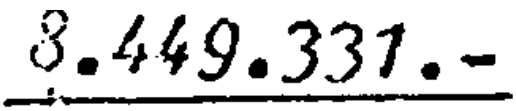


**Figura 4.41:** Campo cédula de identidad para la parte de ficha 1 de la imagen 46 del rollo 570.

Para evitar probar todos los patrones en todas las posiciones como se hizo al clasificar las partes de ficha en la sección 4.2.4, primero se separan los dígitos (y signos de puntuación) y luego para cada dígito detectado se busca cual es el patrón con mayor coincidencia.

Para separar los dígitos se utiliza el vector de suma de intensidades. En esta ocasión, por columnas. Se busca detectar los espacios en blanco entre cada dígito. Estos espacios van a darse donde en el vector de la suma de intensidades contenga mínimos. Para suavizar el vector, antes de hallar los mínimos se aplica un filtro de mediana. Finalmente, se hallan los mínimos usando la biblioteca de señales de scipy. Se pone una restricción de que la distancia entre mínimos debe ser igual o mayor a 20. En caso de que haya dos mínimos a menor distancia, se elimina el valor más grande.

En la Figura 4.42 se encuentra la cédula de la parte de ficha de 1 la imagen 46 del rollo 570 junto con el vector C por columnas suavizado. En dónde se dan los mínimos del vector, es donde hay que separar los dígitos.

El resultado de separar los dígitos por esos mínimos se encuentra en la Figura 4.43. Una vez obtenidos los dígitos, para cada uno se verifica la similitud con los patrones.

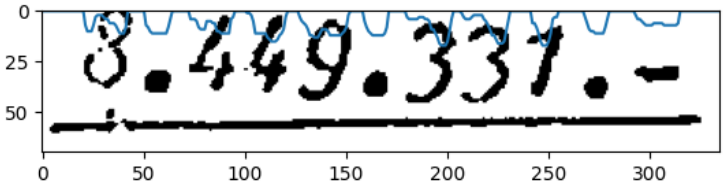



**Figura 4.42:** Campo cédula de identidad para la parte de ficha 1 de la imagen 46 del rollo 570, junto con el vector C suavizado.

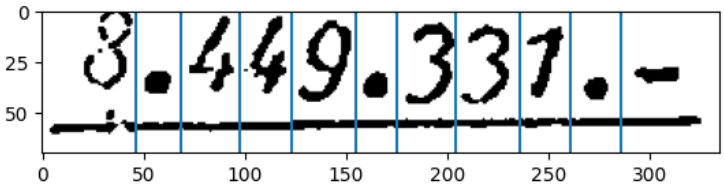



**Figura 4.43:** Campo cédula de identidad para la parte de ficha 1 de la imagen 46 del rollo 570, junto con la separación de dígitos.

Sin embargo, suceden casos en los que las letras no se encuentran lo suficientemente separadas y esta técnica no funciona, devolviendo partes de imagen con dos dígitos.

En la Figura 4.44 hay un ejemplo de una cédula junto con el vector C y las líneas resultantes de encontrar los mínimos. El algoritmo no logra separar correctamente los dígitos. El 0 y el 7 quedan juntos. Observando el vector C se puede ver que entre el primer dígito 0 y el dígito 7 no hay mínimo. Esto se debe a que los dígitos se encuentran muy cerca. La punta de arriba del 7 queda casi tocando el 0 en la parte superior.

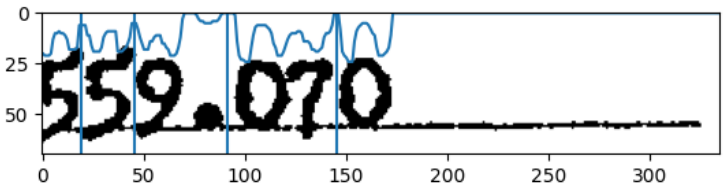



**Figura 4.44:** Campo cédula de identidad para la parte de ficha 3 de la imagen 1369 del rollo 570, junto con la separación de dígitos y vector C.

Otro problema encontrado es que los patrones de los signos de puntuación son muy generales y es posible que tengan una similitud alta con números o partes del fondo. Un guion se parece a un renglón y un punto puede ser un patrón muy general.

Para atenuar este último problema se separan los patrones entre dígitos y puntuación. Al clasificar los recortes de la imagen de la cédula, primero se verifica si existe un patrón de dígito con similitud mayor o igual a $0,885$ para ese recorte. En caso afirmativo, se clasifica ese recorte como de ese patrón y es un dígito. Si no existe un patrón de dígito con tanta similitud, se busca si existe un símbolo de puntuación con similitud mayor o igual a $0,885$. En caso de no existir, se deja ese recorte como de contenido desconocido. El objetivo de realizar la clasificación de esta manera es priorizar la detección de dígitos sobre puntuación.

El gran problema de esta técnica es que los patrones extraídos pueden no ser suficiente para cubrir todos los dígitos de todos los rollos. En el caso del texto escrito a mano, es poco probable que un mismo patrón de un dígito a mano funcione para dos partes de ficha diferentes por más que sean escritos por la misma persona. Además, existen fichas ilegibles incluso para las personas debido a ruido o problemas de adquisición.

Es importante notar también que la velocidad a la que se procesa una imagen con esta técnica depende de la cantidad de patrones. Al aumentar la cantidad de patrones, la velocidad baja.

En la Figura 4.45 se puede ver un ejemplo de una cédula cuyos dígitos son separados e identificados correctamente. En la figura se encuentra también el vector C y la posición donde se cortan los dígitos marcada. Contrario a este caso, se puede ver en la Figura 4.46 una cédula donde el algoritmo no funciona. El dígito 7 se encuentra un poco borrado, lo que genera un mínimo en el vector C y por ende el algoritmo termina cortando el dígito por la mitad.

Luego de recolectar una gran cantidad de patrones (en el orden de los 150) y observar que el desempeño de esta técnica no mejoraba lo suficiente, se decide descartarla y utilizar en su lugar solamente técnicas de aprendizaje automático.

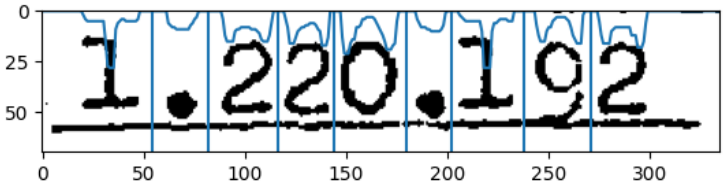



**Figura 4.45:** Campo cédula de identidad para la parte de ficha 1 de la imagen 412 del rollo 570, junto con la separación de dígitos y vector C, los dígitos son identificados correctamente por el algoritmo.

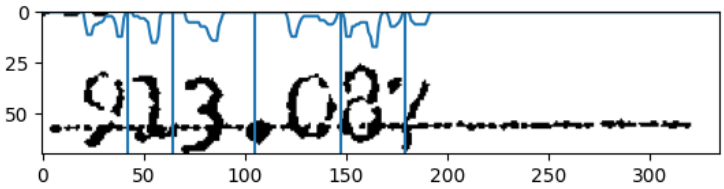



**Figura 4.46:** Campo cédula de identidad para la parte de ficha 0 de la imagen 59 del rollo 570, junto con la separación de dígitos y vector C, los dígitos no se identifican correctamente por el algoritmo.

### 4.5.2. Reconocimiento de texto

Como fue mencionado en la sección 2.1.4, para el reconocimiento de texto se utilizan dos herramientas de OCR: Tesseract y Calamari. A continuación se explican los detalles de su uso.

Wick et al. 2020, los creadores de Calamari, obtuvieron buenos resultados aplicando *finetuning*. Es por esto por lo que se decide utilizar esta técnica con ambos OCR persiguiendo el objetivo de obtener buenos resultados utilizando pocos datos etiquetados.

Se cuenta con un conjunto de datos etiquetado que consiste en líneas de texto de otros rollos pertenecientes al *Archivo Berrutti* que no contiene líneas del fichero de la O.C.O.A. Las etiquetas se obtuvieron mediante Luisa «LUISA», s.f. uno de los proyectos dentro de Cruzar.

Luisa es una plataforma colaborativa creada para transcribir documentos del *Archivo Berrutti*. En ella, voluntarios acceden y transcriben líneas de imágenes pertenecientes al archivo. Estas imágenes fueron previamente procesadas para detectar

y recortar las líneas de texto.

El conjunto de datos accedido consta de tres conjuntos, uno de entrenamiento, uno de validación y uno de evaluación. Cuentan con 21.056, 6.016 y 3.008 imágenes respectivamente.

La metodología de trabajo seguida consistió en entrenar un modelo de Calamari y uno de Tesseract para este conjunto de datos y luego ajustar esos modelos a los datos obtenidos del índice y de las fichas, lo cual requirió etiquetar ejemplos manualmente.

La métrica que se utiliza para evaluar el desempeño de los OCR es *Character Error Rate* (error de reconocimiento de caracteres), abreviada como CER. Esta medida se define como la distancia de edición entre dos cadenas de caracteres, normalizada por el largo máximo. En la Ecuación 4.4 se encuentra la definición para dos cadenas $s_1$ y $s_2$ (Wick et al. 2020), donde $ed$ representa la distancia de edición. En caso de estar evaluando para un conjunto, se toma el promedio.

Esta medida es 0 si las dos cadenas contienen exactamente los mismos caracteres y es 1 en caso de que sean todos diferentes.

$$CER(s_1, s_2) = \frac{ed(s_1, s_2)}{\text{máx}(|s_1|, |s_2|)} \tag{4.4}$$

Los modelos obtenidos de entrenar desde cero con los datos etiquetados en Luisa tuvieron en el conjunto de evaluación, un CER de 0,25 para Calamari y un CER de 0,28 para Tesseract.

Para etiquetar las imágenes, se desarrolla una pequeña aplicación que lee las imágenes desde un directorio y las muestra en un navegador web que contiene un campo de texto para escribir la transcripción junto con botones para enviar la transcripción o etiquetar la imagen como en blanco o que contiene texto ilegible para evitar su uso en el entrenamiento del OCR. Las etiquetas se almacenan en una base de datos MongoDB junto con metadatos que describen el origen del texto. A la hora de entrenar, se leen las etiquetas de la base de datos y se generan los corres-

pondientes archivos con el texto que los OCR utilizan como entrada para entrenar. Una explicación sobre la base de datos generada se encuentra en el Anexo 2.

En la Figura 4.47 se encuentra una captura de pantalla de la interfaz de la aplicación de etiquetado.


**Figura 4.47:** Aplicación desarrollada para etiquetar datos.

Dado que la tipografía utilizada no es uniforme en todas las partes de ficha, el índice, y los números de ficha, se toma la decisión de entrenar varios modelos de OCR y utilizar cada uno en el tipo de texto que fue entrenado. Esta estrategia permitió entrenar y evaluar por separado el OCR para cada conjunto de datos lo que permitió experimentar sin la necesidad de tener datos etiquetados en todos los conjuntos de datos de antemano.

A continuación, se listan los campos de las partes de ficha o columnas del índice en las cuales se etiquetan datos:

- Columna de nombres del índice.
- Columna de número de ficha del índice.
- Campo nombre y campo apellido de las fichas.
- Campo cédulas en las fichas.
- Número de ficha extraído de las fichas.

Tomando como base los modelos entrenados con los datos de Luisa, se entrenó un modelo de Calamari y uno de Tesseract para cada uno de los conjuntos de datos etiquetados realizando *finetuning*.

El modelo entrenado con los nombres del índice se utilizó además para las columnas de observaciones y organización. De manera similar, el modelo entrenado para el número de ficha del índice se utilizó para las columnas que contenían el número en la organización y número de carpeta, el modelo entrenado con los nombres de las fichas se utilizó para todos los campos que no son numéricos en las fichas y finalmente, el modelo entrenado con cédulas se utilizó para todos los campos numéricos de las fichas. La decisión de entrenar un modelo aparte para los números extraídos de las fichas se debe a que estos presentan una tipografía diferente tanto a las cédulas como a los números encontrados en el índice, además de tener ruido de fondo, ya que es posible que se superpongan con texto o con la estructura de la plantilla.

Los modelos de OCR se ejecutaron en un equipo con un i9 de onceava generación, 16 GB de memoria RAM y una tarjeta gráfica Geforce RTX 3060M. Ejecutar ambos OCR en todos los campos extraídos de una parte de ficha demora en promedio 5,94 segundos. Sumando el tiempo promedio para extraer los campos de una ficha mencionado en la sección 4.2.5, se obtiene que el tiempo promedio de tratamiento de una parte de ficha es de 158,94 segundos. Los textos resultantes se almacenaron en una base de datos MongoDB (ver el Anexo 2 para encontrar una explicación de cada colección).

### 4.5.3. Corrección de resultados

Mediante el uso de diccionarios de palabras es posible corregir los resultados del OCR. En particular, se aplicó corrección a los resultados para el índice, en las columnas de organización y nombre.

En el caso de la columnas de organización, primero fue necesario generar un diccionario mediante lectura de los índices. Estas columnas contienen siglas correspondientes a distintas organizaciones. En total, se identificaron 9 organizaciones diferentes: PC, PCR, MLN, UJC, PVP, GAU, FRT, ROE y OPR. Para corregir cada resultado obtenido, se calcula la distancia de edición a todas las palabras del diccionario y en caso de hallar una palabra con distancia menor o igual a 1 entonces se corrige el resultado y se guarda en la base de datos, manteniendo el resultado ori-

ginal. El uso de una distancia de edición tan pequeña solo permite corregir errores menores. No es posible utilizar una distancia de edición mayor ya que podría haber conflictos entre las palabras del diccionario al ser siglas de pocos caracteres. Por ejemplo, si se obtiene como resultado para una organización ABR, se encuentra a distancia 2 tanto de OPR como de PCR y entonces no es posible elegir entre uno de los dos.

Para la corrección de apellidos en el índice, se cuenta con un diccionario que contiene 25,057 apellidos, utilizado previamente en otras investigaciones de Cruzar. En este caso, el proceso de corrección difiere del utilizado para las columnas de organización. Primero, el resultado del OCR se separa por los espacios. Las palabras que son nombres se dejan sin modificar. En este caso es posible identificar las palabras que son nombres ya que los apellidos se encuentran todos en mayúsculas y los nombres solo tienen la primera letra mayúscula. Para los apellidos, en caso de pertenecer al diccionario no se realiza ningún cambio. Si el apellido no pertenece al diccionario, entonces se sustituye por el apellido del diccionario que se encuentre a menor distancia de edición. Se almacena en la base de datos tanto la versión original como la corregida.

Es posible que haya apellidos que no se encuentren en el diccionario, de ahí la importancia de mantener el resultado original de los OCR. Esta corrección se realizó tanto para lo obtenido con Calamari como lo obtenido con Tesseract. Nuevamente, se almacenan los resultados en una base de datos MongoDB.

## 4.6. Visualización de resultados

Con el objetivo visualizar los resultados, se desarrolla una aplicación web de tipo cliente servidor. Al acceder desde un navegador, se cuenta con un listado de buscadores. Gracias a la redundancia de datos, hay varias maneras de buscar una persona. Un primer método consiste en buscar en el índice y luego mediante el número de ficha buscar si se detectaron las partes de ficha asociadas a esa persona. La alternativa, es buscar directamente en los campos de nombre, apellido y cédula en la parte frontal de la ficha. Además, es posible buscar por número de ficha para obtener las partes de ficha asociadas a ese número. Las búsquedas soportadas se

listan a continuación.

- Búsqueda por apellidos y nombres en el índice.
- Búsqueda por número de ficha en el índice.
- Búsqueda por número de ficha extraído de la parte de ficha.
- Búsqueda por número de cédula.
- Búsqueda por apellidos en las partes de ficha.

Al realizar una búsqueda por texto, cada palabra del texto de búsqueda debe encontrarse a una distancia de edición menor o igual a $x$ (configurable desde el navegador) de por lo menos una palabra del resultado del OCR para que esa línea del índice o parte de ficha pertenezca al resultado de la búsqueda. Por ejemplo, si la distancia de edición máxima es 1 y se realiza la búsqueda 'PÉREZ JUAN' entonces el resultado del OCR 'PÉRREZ GONZALEZ GUAN' será incluido en el resultado, ya que la distancia de edición de 'PÉREZ' con 'PÉRREZ' y de 'JUAN' con 'GUAN' es 1 en ambos casos. Esta manera de búsqueda además de mitigar posibles errores del OCR permite buscar parcialmente, sin necesidad de conocer los dos nombres y dos apellidos de la persona. Además, antes de calcular la distancia de edición se convierten tanto el texto de búsqueda como el resultado del OCR todo a minúsculas y se eliminan tildes para que sea más sencilla la búsqueda para el usuario.

En el caso de buscar por apellido dentro del índice, aprovechando que se sabe qué apellidos están en cada rollo debido a que están ordenados alfabéticamente, se decide por defecto limitar la búsqueda a los rollos que contienen ese apellido. Esto logra reducir sustancialmente el tiempo de búsqueda. Además, se incluye la posibilidad de desactivar esta opción. Esto es útil si en lugar de buscar por primer apellido se desea buscar por el segundo apellido.

Al realizar una búsqueda por nombre en el índice, los resultados se muestran en una tabla que contiene el rollo, la imagen, la línea y el resultado de aplicar Calamari y Tesseract sobre la imagen de la columna nombre de esa línea. En la Figura 4.48 se puede ver el resultado de búsqueda para 'PEREZ JUAN'.

PEREZ JUAN | 0 | Buscar | Búsqueda libre

Distancia de edición: 0

| Rollo | Imagen | Linea | OCR Calamari | OCR Tesseract |
|---|---|---|---|---|
| r0608 | r0608_0029.tif | 29 | PEREZ RJSTAMANTE Juan Carlos | PEREZ RJSTAMANTE Juan Carlos |
| r0608 | r0608_0030.tif | 24 | PEREZ FERREIRA Juan Manuel Isaa | PEREZ FERREIRA Juan Manuel Isaaco |
| r0608 | r0608_0031.tif | 24 | PEREZ Juan A | PEREZ Juana A |
| r0608 | r0608_0032.tif | 13 | PEREZ ORTEGA Juan Carlos | PEREZ ORTEGA Juan Carlos |
| r0608 | r0608_0032.tif | 19 | PEREZ PEREZ Ana María | PEREZ PEREZ Ana María |
| r0608 | r0608_0032.tif | 20 | PEREZ PEREZ Beniamín Julio | PEREZ PEREZ Beniamín Jullo |
| r0608 | r0608_0032.tif | 21 | PEREZ PEREZ Diego Raúl | PEREZ PEREZ Diego Raúúl |
| r0608 | r0608_0032.tif | 22 | PEREZ PEREZ Dina | PEREZ PEREZ Diiina |
| r0608 | r0608_0032.tif | 23 | PEREZ PEREZ Humberto | PEREZ PEREZ Humberrto |
| r0608 | r0608_0032.tif | 24 | PEREZ PEREZ Jupiter | PEREZ PEREZ Jupiter |

**Figura 4.48:** Visualización de resultados al buscar en el índice.

Al seleccionar una fila, se accede a una página que muestra la línea completa, los resultados del OCR para cada columna y las partes de ficha cuyo número de ficha extraído coincide con el número de ficha del índice. Por ejemplo, en la Figura 4.49 se encuentra el detalle de una de las filas de la búsqueda anterior. En este caso particular, el número de ficha es 22510, que fue detectado correctamente por ambos OCR utilizados. Además, se detectaron 6 partes de ficha asociadas a dicho número que se pueden ver abajo de la tabla con los resultados del OCR. Las primeras 5 son partes de atrás y la última es una parte frontal.

22510 PEREZ ORTEGA Juan Carlos P C 1275 4038

| Columna | OCR Calamari | OCR Tesseract |
|---|---|---|
| Número ficha | 22510 | 22510 |
| Nombre | PEREZ ORTEGA Juan Carlos | PEREZ ORTEGA Juan Carlos |
| Alias | | ... |
| Organización 1 | PG | PCC |
| Número organización 1 | 1275 | 17275 |
| Organización 2 | | |
| Número organización 2 | | |
| Número carpeta | SO38 | ADOSA33 |
| Observaciones | | |

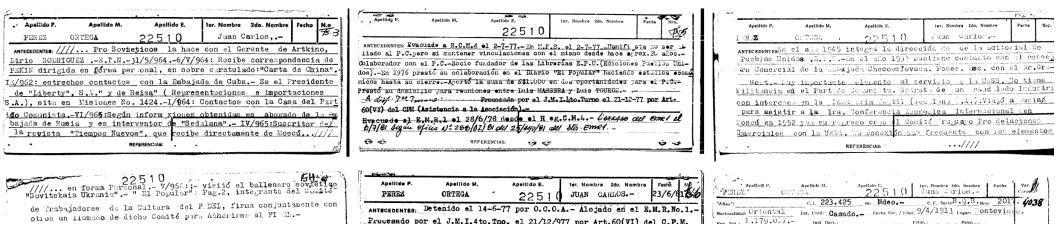


**Figura 4.49:** Visualización de detalle de resultado al buscar por apellido y nombre en el índice.

En caso de seleccionar una de las partes de ficha, se accede a una vista detallada donde se puede apreciar la imagen más grande y en caso de ser una frontal si esta fue clasificada se encuentra también una tabla de resultados de los OCR para cada campo extraído y una imagen que muestra donde se detectaron los patrones junto con los campos extraídos. En la Figura 4.50 se muestra parte de la tabla de resultados para la única parte frontal de la búsqueda utilizada como ejemplo. En la Figura 4.51 se puede observar dónde se detectaron los patrones y los campos extraídos. Hay algunos campos que no se detectaron correctamente, como dirección trabajo y nombre esposa que se encuentran en la parte inferior derecha. Además, se identifica otro problema en el cual el campo ocupación hay una segunda dirección, que incluso alcanza a superponerse con el campo dirección trabajo y la ocupación de esta persona (abogado) queda por abajo del campo.

| Campo | OCR Calamari | OCR Tesseract |
|---|---|---|
| Apellido E | PEREZ C | |
| Apellido Materno | PEREZ C | |
| Apellido Paterno | PEREZ C | BIRD I |
| Primer nombre | JRa JrlOs. | Jua Arlos. |
| Segundo nombre | RRAA IOR.- | Juan irlos. |
| Fecha | | |
| Alias | | |
| CI | 223.425 | 223.425 |
| De | Mdeo.- | |
| CC Serie | DD. | |
| Nro | 501 | :: |
| Nacionalidad | OEntal | Grlental |
| Estado civil | Casado.- | Casado. |
| Fecha nacimiento/Edad | | GALOI |

**Figura 4.50:** Tabla con los resultados de ambos OCR a una parte frontal de ficha.

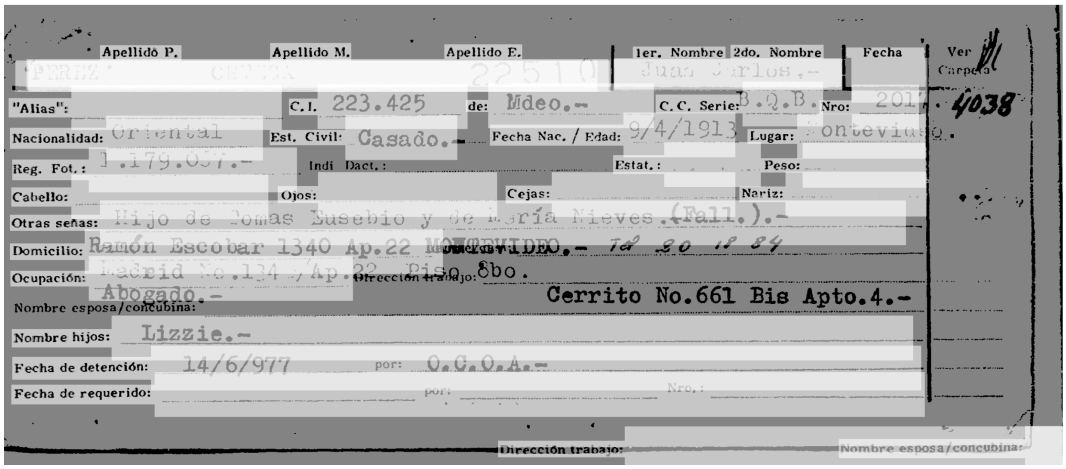



**Figura 4.51:** Parte de ficha junto con detección de campos.

Al realizar una búsqueda en las partes de ficha por número de cédula o apellido, la visualización de resultados obtenidos es similar al realizar una búsqueda en el índice. Sin embargo, en este caso no hay columna de línea. Al seleccionar una fila dentro de los resultados, se accede directamente a la vista detallada de la parte de ficha en la cual se encuentra la tabla que presenta los resultados del OCR y la imagen que muestra donde se detectaron los campos.

Por último, la búsqueda por número de ficha extraído muestra dos tablas de resultados. Una para cada método de extracción del número de ficha. Al seleccionar una de las filas, se accede directamente a la vista detallada de la parte de ficha que contiene la imagen, los resultados del OCR y la imagen con la detección de los campos. En la Figura 4.52 se muestran los resultados al buscar la ficha número 19852. Es importante destacar que se obtuvieron resultados distintos con ambos métodos. Además, se observa que para la misma imagen, r0603_2060_3.png según el método de extracción Tesseract devolvió diferentes resultados. Una posible explicación de esto podría ser que las imágenes resultantes de ambos métodos son diferentes.

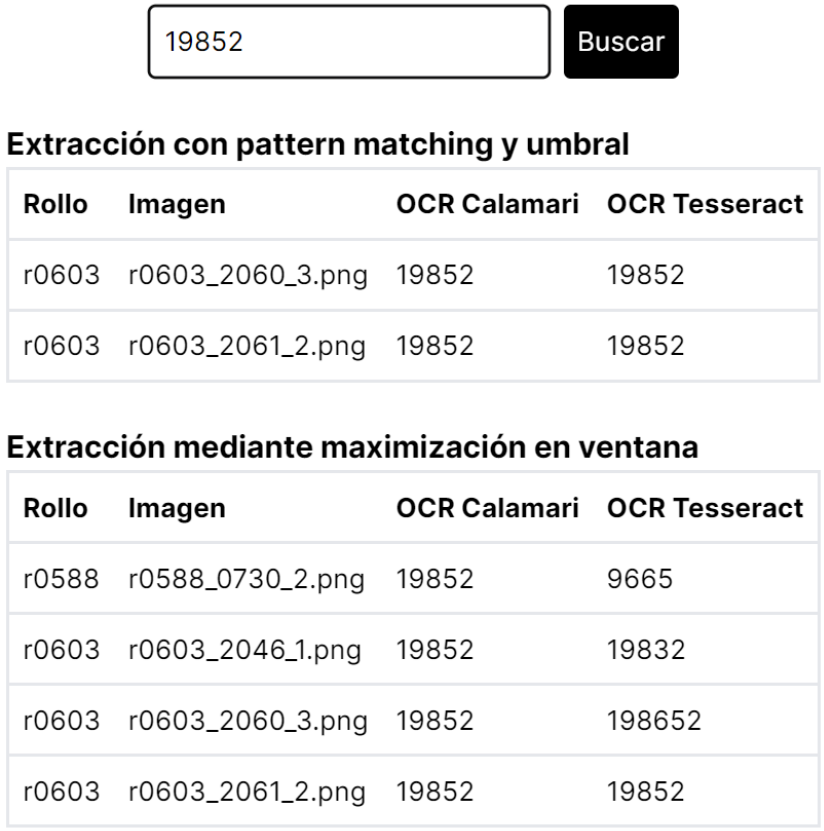



**Figura 4.52:** Resultados de búsqueda por número de ficha.

Además de las formas de buscar implementadas, existen otras formas no desarrolladas. Por ejemplo, búsqueda por número de ficha pero dentro del índice o por las columnas de organización que pertenecen al índice. Otra posibilidad es buscar dentro de las partes de ficha utilizando otros campos que no sean el de apellido o cédula. Sin embargo, las formas de búsqueda desarrolladas son adecuadas como una primera aproximación a un visualizador de resultados.

# Capítulo 5

# Evaluación y resultados

En este capítulo, se presenta cómo fue evaluada la solución y se analizan los resultados obtenidos de aplicar la metodología descrita en el capítulo 4. Las secciones de este capítulo van a corresponderse con las del capítulo 4.

## 5.1. Extracción de partes de fichas de las hojas

El objetivo de esta sección es realizar una evaluación y análisis del método desarrollado para separar las partes de ficha en una hoja y almacenarlas por separado.

Se considera que una parte de ficha fue cortada correctamente si en la imagen resultante se encuentra contenida totalmente la parte de ficha, no hay partes de las fichas vecinas y la plantilla se encuentra derecha. En términos generales, para una parte frontal esto significa que en la imagen se encuentra desde el número de ficha en la parte superior hasta el campo 'Fecha de requerido' en la parte inferior. Para una parte de atrás, significa que se encuentre desde el número de ficha en la parte superior hasta el texto de más abajo, que en muchos casos es la palabra 'REFERENCIAS'. Además, no tiene mucho sentido evaluar el resultado en partes de ficha que son huellas digitales o solo una fotografía ya que el método fue desarrollado pensando en las partes de adelante y atrás de las fichas.

El requerimiento de que la plantilla se encuentre derecha se mide a simple vista. Una rotación pequeña no afecta los pasos siguientes. Además, el método de enderezar utilizado puede verse afectado por ruido en la imagen.

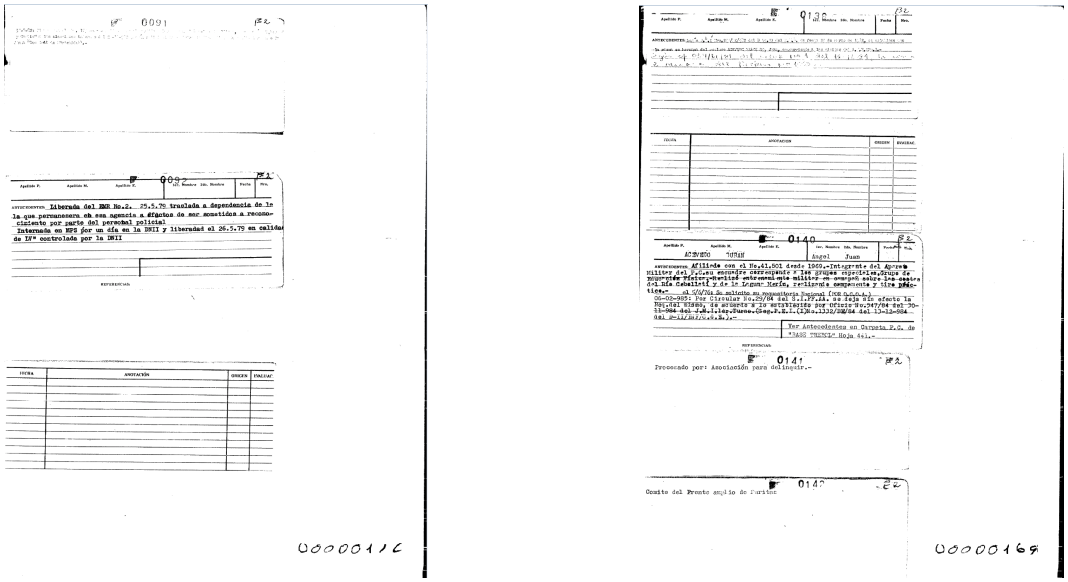
Para evaluar los resultados, se considera un conjunto seleccionado al azar del rollo 570 compuesto por 50 imágenes, 10 de cada categoría (10 con una parte de ficha sola, 10 con 2 partes de ficha, 10 con 3 partes de ficha, 10 con 4 partes de ficha y 10 con 5 partes de ficha). A continuación, se ejecuta el algoritmo en estas hojas y se realiza una verificación manual del resultado. Contabilizando la cantidad de partes de ficha detectada por hoja y si las mismas fueron extraídas correctamente. En total, hay $10 * (1+2+3+4+5) = 150$ partes de ficha en las hojas seleccionadas.

Se detectaron únicamente dos instancias en las que se encontró una cantidad incorrecta de fichas. En una de ellas, se esperaban cinco partes de ficha, pero solo se detectaron tres, como se muestra en la Figura 5.1a. Por otro lado, en la segunda instancia, la hoja en cuestión debía contener tres partes de ficha, pero solo se detectaron dos, tal como se ilustra en la Figura 5.1b.

En ambos casos, las partes de ficha no extraídas corresponden a partes de atrás, que no cuentan con una plantilla definida y además tienen pocas líneas. Por lo tanto, resulta comprensible que el algoritmo no las haya detectado. En total, se detectaron 147 partes de ficha sobre un total de 150 disponibles. Esto representa un 98 % de partes de ficha detectadas correctamente.

En relación a la extracción de las partes de ficha, todas las detectadas fueron extraídas correctamente, a excepción de siete casos. Estas siete excepciones corresponden exclusivamente a hojas que contienen cinco partes de ficha. Además, todas estas extracciones incorrectas pertenecen solo a cuatro hojas. En el caso de las hojas con cinco partes de ficha, como las mismas se encuentran con muy poca o nula separación, en caso de encontrar una extracción errónea es probable que haya más extracciones erróneas. De las 147 partes de ficha detectadas, se extrajeron 140 correctamente, representando un 95 % de partes de fichas detectadas extraídas.

Considerando que había 150 partes de ficha y se extrajeron correctamente 140 entonces se tiene que un 93 % del total de partes de ficha se extrajeron correctamente.



**(a)** Hoja 116 del rollo 570.

**(b)** Hoja 169 del rollo 570.

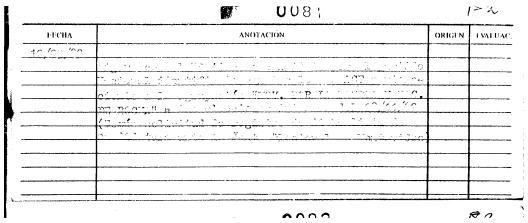
**Figura 5.1:** Únicas hojas en las que el algoritmo falló en detectar la cantidad de partes de ficha correctamente.

En la Figura 5.2 se pueden observar todas las partes de ficha extraídas incorrectamente junto con una explicación de cuál fue el problema en cada caso.

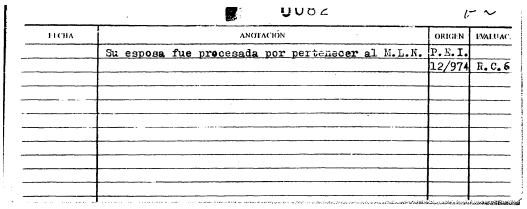
Una vez ejecutado el algoritmo en todos los rollos, se obtuvieron 61.476 partes de ficha. En la Figura 5.3 se presentan dos gráficas de barras. En la de la izquierda, se puede ver el total de hojas según cuantas partes de fichas fueron extraídas y en la derecha el total de partes de ficha (es decir, la gráfica de la derecha surge de multiplicar por la cantidad de partes de ficha).

Lo primero que llama la atención es la escasa cantidad de hojas que contenían cinco fichas, siendo únicamente 159. Durante la prueba previa llevada a cabo, uno de los casos donde el algoritmo cometió errores fue cuando la cantidad de partes de ficha era cinco en una hoja. Aun teniendo en cuenta que el error es mayor para esta cantidad de partes de ficha, el número de hojas que se detectaron cinco partes de fichas es sumamente reducido. Si además se considera que los problemas de extracción de la prueba de evaluación realizada surgieron en su totalidad cuando la hoja tenía cinco fichas, este resultado es muy alentador.

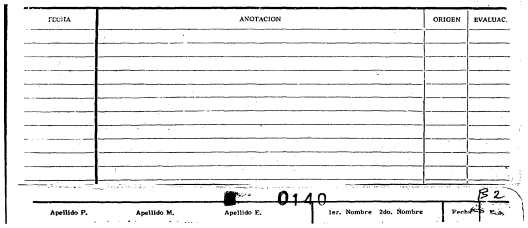
Si se piensa en las razones prácticas, la colocación de cinco partes de ficha en una sola hoja resultaba en casi un solapamiento entre todas ellas, por lo tanto, la



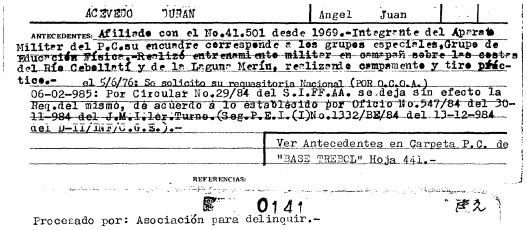

**(a)** Primera parte de ficha de la hoja 107 del rollo 570. En esta ocasión, parte del número de la ficha quedó cortado y aparece una pequeña porción de la parte de ficha de abajo.



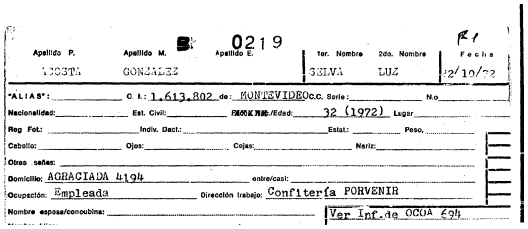

**(b)** Segunda parte de ficha de la hoja 107 del rollo 570. El problema en este caso es que el número de ficha está un poco cortado.



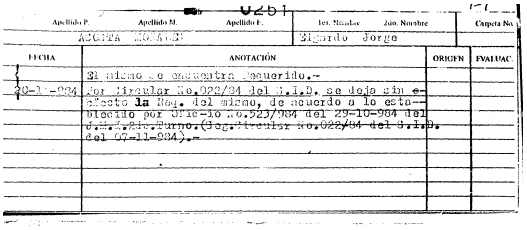

**(c)** Segunda parte de ficha de la hoja 168 del rollo 570. En la parte inferior aparece una parte de la siguiente parte de ficha en la hoja.



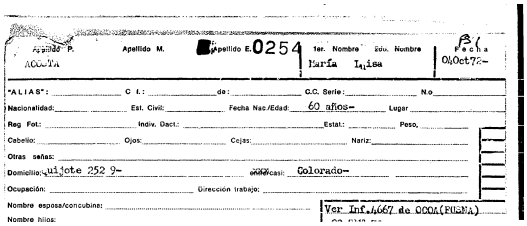

**(d)** Tercera parte de ficha de la hoja 169 del rollo 570. Sucede algo similar al caso anterior, aparece parte de la siguiente parte de ficha en la hoja.




**(e)** Segunda parte de ficha de la hoja 253 del rollo 570. En este caso, el campo 'Nombre hijos' se encuentra por afuera del recorte.

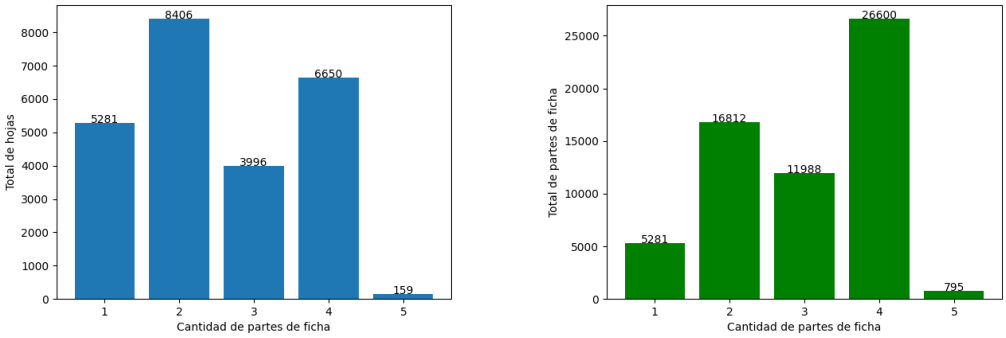



**(f)** Primera parte de ficha de la hoja 277 del rollo 570. Aparece cortado el número en la parte superior.




**(g)** Cuarta parte de ficha de la hoja 277 del rollo 570. En la esquina inferior derecha, hay una línea de texto que se encuentra cortada.

**Figura 5.2:** Partes de ficha recortadas incorrectamente. Cada imagen contiene la explicación de porque está mal recortada.




**(a)** Distribución de la cantidad de fichas por hoja.

**(b)** Total de fichas extraídas según cuantas había en la hoja.

**Figura 5.3:** Gráficas de resultados de la extracción de partes de ficha.

persona que realizó la microfilmación debe haber intentado evitar esta situación. No obstante, la verdadera causa de por qué la cantidad de partes de ficha por hoja es tan diversa es desconocida.

En la gráfica de la derecha, se observa que en el total de partes de ficha extraídas, el 43 % corresponde a hojas con cuatro partes de ficha, un 27 % de las fichas fueron extraídas de una hoja con dos partes de ficha, un 19 % de una hoja con tres partes de ficha, un 8 % estaban solas en la hoja y finalmente solamente un 1 % corresponden a hojas con cinco partes de ficha.

En la Tabla 5.1 se muestra la cantidad de partes de ficha extraídas de cada rollo. Al comparar estos resultados con la cantidad de fichas por rollo en la Tabla 3.1, se observa que, en general, se extraen un poco menos de dos partes de ficha por cada ficha. Esto pone de manifiesto la presencia de algunos errores durante el proceso de extracción, dado que se espera, en general, que existan al menos dos partes de ficha por cada ficha, una frontal y una de atrás.

## 5.2. Clasificación de partes de ficha

Para evaluar el resultado de la clasificación se genera un conjunto de datos. Se seleccionan manualmente 10 imágenes de cada clase de parte frontal, 20 ejemplos

**Tabla 5.1:** Cantidad de partes de ficha extraídas de cada rollo.

| Rollo | Cantidad partes de ficha |
|---|---|
| 570 | 5.099 |
| 584 | 5.329 |
| 585 | 6,354 |
| 587 | 3.430 |
| 588 | 4.023 |
| 594 | 5.317 |
| 599 | 4.452 |
| 603 | 5.777 |
| 607 | 2.066 |
| 608 | 4.847 |
| 612 | 4.317 |
| 613 | 4.826 |
| 614 | 1.722 |
| 615 | 3.917 |

de partes de atrás y 10 imágenes que son fotografías o huellas dactilares, que pertenecen a la categoría otros. En total hay 130 partes de ficha.

Las imágenes fueron seleccionadas del rollo 570 con excepción de algunas para la clase 3 de parte frontal que tuvieron que tomarse también de los rollos 584 y 585 debido a que es una clase con una frecuencia muy baja.

A estas imágenes seleccionadas, se le aplica el algoritmo de clasificación de la misma manera que se realizó con todas las partes de ficha, en las mismas condiciones.

En la Figura 5.4 se presenta la matriz de confusión con los resultados. En general, los resultados son bastante alentadores. Se observan pocas equivocaciones. Ocho clases exhiben resultados perfectos. Para las clases restantes, hay una parte de atrás que es incorrectamente clasificada dentro de la categoría otros. Tres partes de ficha de la clase 5 fueron incorrectamente clasificadas dentro de la categoría otros. Una parte de ficha de la clase 7 y una de la clase 10 fueron clasificadas incorrectamente dentro de otra clase.

Analizando las imágenes clasificadas incorrectamente para intentar comprender qué sucedió, en el ejemplo de la parte de atrás clasificada dentro de la categoría

otros, se observa que la palabra REFERENCIAS de la plantilla, que es el patrón utilizado para clasificar, quedó cortado al recortar la parte de ficha y por lo tanto no fue detectado.

Para el ejemplo de la clase 7, la parte de ficha tiene una calidad muy mala y posiblemente por eso no haya sido clasificada correctamente.

En el caso de la parte de ficha mal clasificada de la categoría 10 y las tres partes de ficha clasificadas incorrectamente de la clase 5 no se observa ningún problema en las imágenes a simple vista.

El accuracy global obtenido en este experimento es de 95 %, lo cual se considera un resultado satisfactorio.

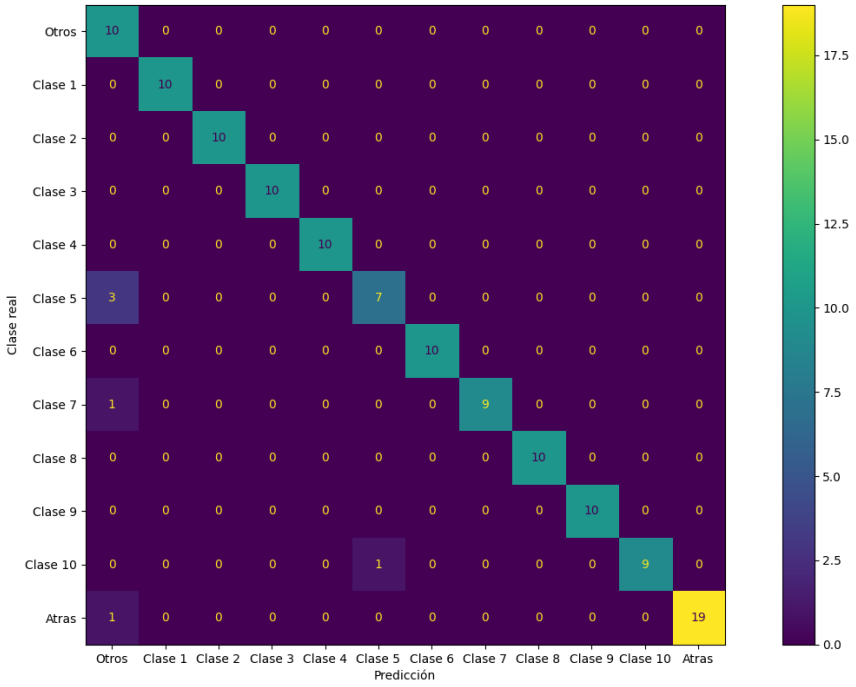



**Figura 5.4:** Matriz de confusión de los resultados del experimento de clasificación.

En relación a los resultados en todas las partes de ficha, en la Figura 5.5 se presenta un gráfico de barras que muestra la cantidad de partes de ficha clasificadas

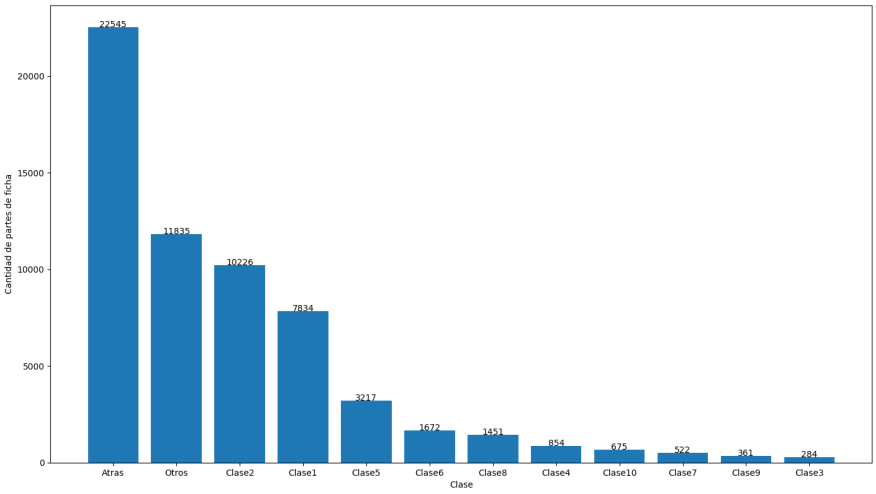



**Figura 5.5:** Resultados de clasificación de todos los rollos. Cantidad de partes de ficha por clase.

en cada clase, ordenadas de clase más frecuente a clase menos frecuente. Como era de esperar, la clase más frecuente corresponde a partes de atrás de ficha. La segunda categoría más frecuente es la clase 'otros'. Este resultado es previsible ya que esta clase incluye fotografías, huellas dactilares y las partes frontales y de atrás que no pudieron ser clasificadas debido a problemas de calidad en la imagen.

En cuanto a las partes frontales, las clases más comunes, la 2 y la 1, incluyen un número considerablemente superior de ejemplos, que supera la suma de todos los ejemplos de las clases restantes. Si se suman todas las partes frontales detectadas, se obtiene un total de 27.096. Por otro lado, la cantidad de partes de atrás detectadas, 22.545 es notablemente inferior. Esto podría sugerir que hay bastantes partes de atrás que fueron clasificadas dentro de la categoría otros o que hay muchas fichas sin parte de atrás.

Cómo fue mencionado en el capítulo 3, hay 31.435 individuos. Entonces, se puede afirmar que hay al menos 4.339 partes frontales que no fueron extraídas y clasificadas, ya que se clasificó un total de 27.096 partes frontales. No obstante, existen casos de personas con más de una parte frontal. Esta cifra representa aproximadamente un 14 % del total de personas.

Al analizar manualmente las imágenes que fueron clasificadas dentro de la categoría otros, se observa una considerable cantidad de partes frontales y partes de atrás. Esto se debe a que no hubo suficientes patrones con un coeficiente de similitud que superara el umbral por problemas en las imágenes.

Se han detectado también algunos casos de partes frontales clasificadas en la clase incorrecta. Una consecuencia de este problema es que para esas partes frontales los campos no van a extraerse correctamente.

## 5.3. Extracción de número de ficha

En esta sección, se lleva a cabo una evaluación del rendimiento del método desarrollado para extraer el número de ficha de las partes frontales y de atrás. Dado que el número se extrajo dos veces, utilizando métodos diferentes, resulta pertinente realizar una comparación entre ambos enfoques y evaluar tanto su efectividad individual como la viabilidad de utilizar los resultados de ambos en conjunto.

Para crear un conjunto de evaluación, se procede a seleccionar manualmente diversas imágenes. Se busca lograr una variedad en dicho conjunto, incluyendo imágenes en las cuales se espera que los métodos funcionen correctamente. Por ejemplo, imágenes en las que el número no se encuentre superpuesto a ningún otro texto o estructura de la plantilla. Asimismo, se incluirán imágenes en las que sea probable que los métodos fallen, como aquellas en las que el número se encuentre por encima de texto o estructura de la plantilla.

En total, se seleccionan 100 partes de ficha, la mitad de ellas con números de cuatro dígitos y la otra mitad con números de cinco dígitos. Posteriormente, se aplicaron ambos métodos de extracción del número a cada imagen. Al momento de evaluar el resultado, se considera que la extracción fue correcta si en la imagen resultante se encuentra todo el número.

Para el método de *template matching* múltiple y umbral se obtiene que el número fue extraído correctamente en 97 casos de 100 y para el método de maximización en ventana, el número se extrajo correctamente en 89 casos de 100. Además, es

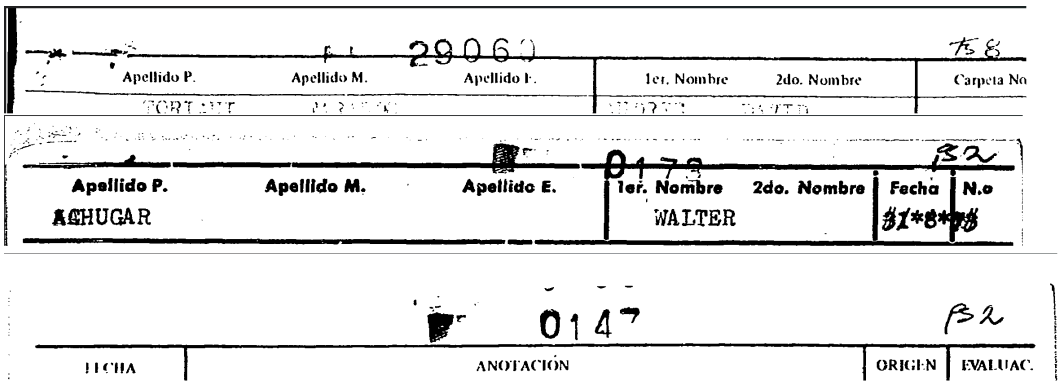



**Figura 5.6:** Fichas cuyo número no fue extraído correctamente por el método de *template matching* y umbral.

importante destacar que no hubo ninguna imagen cuya extracción del número haya fallado con ambos métodos.

En los tres casos en que el número se extrajo incorrectamente con el método de *template matching* y umbral, se observa que en todos los casos hay un dígito que se encuentra parcialmente borrado, lo que pudo ocasionar que no se alcanzara el umbral. En la Figura 5.6 se observa la parte superior de estas tres partes de fichas. En todos los casos el último dígito se encuentra parcialmente borrado.

En el método de extracción utilizando la ventana, en la mayoría de los casos el problema que sucedió es que parte de la estructura de la plantilla tuvo un coeficiente de similitud alto con el patrón asociado al 1, incluso mayor que con el patrón correspondiente al dígito correcto, ocasionando que el dígito quedara por fuera del recorte.

En la Figura 5.7, que contiene dos imágenes, se puede observar un ejemplo de extracción fallido mediante el método de *template matching* y maximización en ventana. En la imagen superior se encuentra el resultado. Le falta el primer dígito pero en la imagen se encuentra parte de la estructura de la plantilla. La imagen inferior enseña toda la parte superior de la parte de ficha. Se puede ver que el dígito no extraído es un dos y se encuentra en una zona con ruido ya que se encuentra por encima de una etiqueta de un campo.

Se concluye que ambos métodos arrojaron resultados satisfactorios a la hora

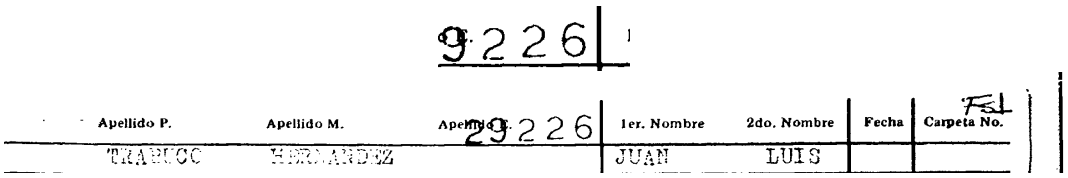



**Figura 5.7:** Ejemplo de parte de ficha con número mal extraído con el método de extracción utilizando *template matching* y maximización en ventana.

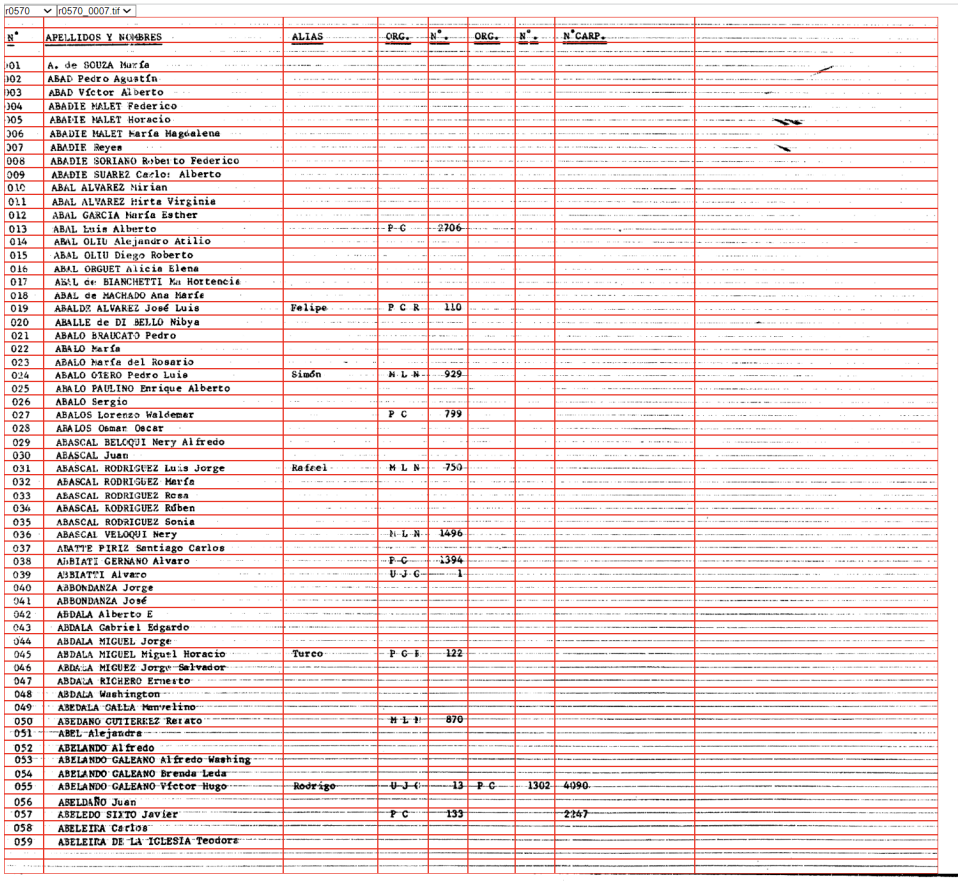
de extraer el número de ficha. El método de extracción mediante *template matching* múltiple y umbral fue superior al de maximización en ventana. Sin embargo, utilizar ambos métodos al mismo tiempo fue una estrategia acertada dado que, en general, cuando uno fallaba, el otro lograba funcionar correctamente. Queda pendiente para la próxima sección la evaluación del resultado obtenido al aplicar OCR a la imagen resultante con los números.

Finalmente, también se tiene la cifra de en cuántas imágenes no se logró un umbral suficiente al buscar el primer dígito, resultando en que no se extraiga número en esas imágenes. Entre todos los rollos, hay solamente un total de 182 imágenes. Al analizar manualmente algunos ejemplos, se observa que en general el número no estaba dentro de la zona de búsqueda y se encontraba por ejemplo, en la parte inferior o era una imagen sin número o era una fotografía sin número (muchas fotografías tienen el número de ficha marcado).

## 5.4. Procesamiento del índice

Para poder evaluar los resultados de dividir las líneas y columnas del índice, se desarrolla una pequeña aplicación web que para cada rollo e imagen del índice, muestra el resultado de cortar las líneas y columnas. En la Figura 5.8 se puede observar esta aplicación con una hoja del rollo 570. En la parte superior se encuentra un selector de rollo y de imagen. Líneas rojas indican donde fue cortada la imagen original.

Se considera que una imagen del índice se encuentra adecuadamente cortada si cada celda contiene la información correcta. Esto es, la información de la celda debe coincidir con la columna a la que pertenece y además, el texto no debe estar



| r0570 | r0570_0007.tif | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| N° | APELLIDOS Y NOMBRES | ALIAS | ORG. | N°. | ORG. | N°. | N°CARP. | |
| 001 | A. de SOUZA María | | | | | | | |
| 002 | ABAD Pedro Agustín | | | | | | | |
| 003 | ABAD Víctor Alberto | | | | | | | |
| 004 | ABADIE MALET Federico | | | | | | | |
| 005 | ABADIE MALET Horacio | | | | | | | |
| 006 | ABADIE MALET María Magdalena | | | | | | | |
| 007 | ABADIE Reyes | | | | | | | |
| 008 | ABADIE SORIANO Roberto Federico | | | | | | | |
| 009 | ABADIE SUAREZ Carlos Alberto | | | | | | | |
| 010 | ABAL ALVAREZ Nirian | | | | | | | |
| 011 | ABAL ALVAREZ Hirta Virginia | | | | | | | |
| 012 | ABAL GARCIA María Esther | | | | | | | |
| 013 | ABAL Luis Alberto | | P C | 2706 | | | | |
| 014 | ABAL OLIU Alejandro Atilio | | | | | | | |
| 015 | ABAL OLIU Diego Roberto | | | | | | | |
| 016 | ABAL ORGUET Alicia Elena | | | | | | | |
| 017 | ABAL de BIANCHETTI Ma Hortencia | | | | | | | |
| 018 | ABAL de MACHADO Ana María | | | | | | | |
| 019 | ABALDE ALVAREZ José Luis | Felipe | P C R | 110 | | | | |
| 020 | ABALLE de DI BELLO Nibya | | | | | | | |
| 021 | ABALO BAUCATO Pedro | | | | | | | |
| 022 | ABALO María | | | | | | | |
| 023 | ABALO María del Rosario | | | | | | | |
| 024 | ABALO OTERO Pedro Luis | Simón | M L N | 929 | | | | |
| 025 | ABALO PAULINO Enrique Alberto | | | | | | | |
| 026 | ABALO Sergio | | | | | | | |
| 027 | ABALOS Lorenzo Waldemar | | P C | 799 | | | | |
| 028 | ABALOS Osmar Oscar | | | | | | | |
| 029 | ABASCAL BELCQUI Nery Alfredo | | | | | | | |
| 030 | ABASCAL Juan | | | | | | | |
| 031 | ABASCAL RODRIGUEZ Luis Jorge | Rafael | M L N | 750 | | | | |
| 032 | ABASCAL RODRIGUEZ María | | | | | | | |
| 033 | ABASCAL RODRIGUEZ Rosa | | | | | | | |
| 034 | ABASCAL RODRIGUEZ Rúben | | | | | | | |
| 035 | ABASCAL RODRIGUEZ Sonia | | | | | | | |
| 036 | ABASCAL VELOQUI Nery | | M L N | 1496 | | | | |
| 037 | ABATTE PIRIZ Santiago Carlos | | | | | | | |
| 038 | ABBIATI GERNANO Alvaro | | P C | 1394 | | | | |
| 039 | ABBIATTI Alvaro | | U J C | 1 | | | | |
| 040 | ABBONDANZA Jorge | | | | | | | |
| 041 | ABBONDANZA José | | | | | | | |
| 042 | ABDALA Alberto E | | | | | | | |
| 043 | ABDALA Gabriel Edgardo | | | | | | | |
| 044 | ABDALA MIGUEL Jorge | | | | | | | |
| 045 | ABDALA MIGUEL Miguel Horacio | Turco | P C R | 122 | | | | |
| 046 | ABDALA MIGUEZ Jorge Salvador | | | | | | | |
| 047 | ABDALA RICHERO Ernesto | | | | | | | |
| 048 | ABDALA Washington | | | | | | | |
| 049 | ABEDALA CALLA Manvelino | | | | | | | |
| 050 | ABEDANO GUTIERREZ Renato | | M L N | 870 | | | | |
| 051 | ABEL Alejandra | | | | | | | |
| 052 | ABELANDO Alfredo | | | | | | | |
| 053 | ABELANDO GALEANO Alfredo Washing | | | | | | | |
| 054 | ABELANDO GALEANO Brenda Leda | | | | | | | |
| 055 | ABELANDO GALEANO Víctor Hugo | Rodrigo | U J C | 13 | P C | 1302 | 4090 | |
| 056 | ABELDAÑO Juan | | | | | | | |
| 057 | ABELEDO SIXTO Javier | | P C | 133 | | | 2247 | |
| 058 | ABELEIRA Carlos | | | | | | | |
| 059 | ABELEIRA DE LA IGLESIA Teodora | | | | | | | |

**Figura 5.8:** Aplicación para visualizar resultados de cortar líneas y columnas del índice.

cortado horizontalmente. En el ejemplo de la Figura 5.8 la hoja ha sido cortada correctamente.

Se seleccionan aleatoriamente 5 imágenes por rollo y se analiza si la extracción de las celdas fue satisfactoria. En total, son 115 imágenes para analizar, y representa un poco más del 10 % del total de imágenes.

Del total imágenes, 98 fueron procesadas correctamente y 17 incorrectamente. Esto representa éxito en el 85 % de los casos. En la Tabla 5.2 se puede observar cuantas hojas fueron extraídas correctamente para cada uno de los rollos.

**Tabla 5.2:** Cantidad de hojas del índice cortadas correctamente de las revisadas.

| # Rollo | # Imágenes cortadas correctamente |
|---|---|
| 570 | 3 |
| 570v1 | 5 |
| 584 | 4 |
| 585 | 5 |
| 585v1 | 5 |
| 587v1 | 4 |
| 588 | 5 |
| 588v1 | 4 |
| 594 | 4 |
| 599 | 5 |
| 599v1 | 4 |
| 599v2 | 5 |
| 603 | 4 |
| 603v1 | 5 |
| 607 | 5 |
| 607v1 | 4 |
| 608 | 2 |
| 608v1 | 3 |
| 612 | 4 |
| 612v1 | 4 |
| 613 | 5 |
| 614 | 5 |
| 615 | 4 |

El rollo 608 fue el que tuvo peor desempeño, ya que solo dos de cinco hojas revisadas lograron extraerse correctamente. Esto puede deberse a mala suerte al elegir aleatoriamente las imágenes a revisar. En la mayoría de los casos hubo cuatro o las

cinco hojas correctamente procesadas, mientras que en solo dos casos se procesaron únicamente tres hojas de manera adecuada.

Para el rollo 608, se decide analizar los resultados de 30 imágenes más, lo que eleva el total de imágenes analizadas a 35 para ese rollo. De este total, en 24 imágenes las filas y columnas del índice fueron correctamente cortadas. Esto representa un 69 % de los casos. Este resultado se encuentra por debajo de lo observado en general pero es superior al resultado obtenido previamente solo analizando cinco imágenes.

## 5.5. Reconocimiento de texto

En esta sección, se exponen los desafíos y resultados obtenidos al aplicar técnicas de OCR a las imágenes. Se entrenaron diversos modelos, utilizando conjuntos de datos distintos. Para cada conjunto, se proporciona una breve descripción, acompañada de las métricas de rendimiento correspondientes. Además, se ofrece una justificación de por qué al final se optó por no utilizar *template matching* para el reconocimiento de cédulas. Finalmente, se realiza una comparación con un OCR comercial de Google.

### 5.5.1. Datos extraídos del índice

En el caso del índice, se contaba con datos de 23 rollos. Una vez ejecutado el algoritmo para separar cada imagen del índice en líneas, se obtiene un total de 61.758 líneas, algunas de ellas en blanco. Al momento de etiquetar imágenes, se seleccionaron aleatoriamente entre todos los rollos.

#### 5.5.1.1. OCR número de ficha del índice

El primer modelo de OCR entrenado en el índice fue con intención de reconocer el número de ficha en cada línea. La cantidad de datos etiquetada fue de 1210

imágenes. Este proceso llevó aproximadamente tres horas. A la hora de entrenar el modelo, 1029 fueron utilizadas en el conjunto de entrenamiento y 181 en el conjunto de evaluación.

En la Figura 5.9 se observa un histograma tanto para el conjunto de entrenamiento como para el conjunto de evaluación que muestra la cantidad de ejemplos en cada rollo. Se observa que en ambos conjuntos hay ejemplos de todos los rollos y además las distribuciones son similares.

En todos los casos, el modelo de base utilizado para comparar es el entrenado en el conjunto de datos de Luisa. En la Tabla 5.3 se observan los resultados obtenidos luego de entrenar el OCR y utilizando el modelo base. En este caso, el modelo base ya daba unos resultados bastante buenos en ambos casos, siendo Calamari levemente superior a Tesseract. Luego del entrenamiento, se observa una mejora sustancial del rendimiento de ambos OCR, obteniendo en los dos casos un CER de $0,0033$ en el conjunto de evaluación.

**Tabla 5.3:** Resultados OCR para el número de ficha en el índice.

| | CER Entrenamiento | CER Evaluación |
|---|---|---|
| Calamari baseline | 0,0392 | 0,0385 |
| Calamari ajustado | 0,0011 | 0,0033 |
| Tesseract baseline | 0,0656 | 0,0549 |
| Tesseract ajustado | 0,00048 | 0,0033 |

En la Figura 5.10 se puede observar un ejemplo perteneciente al conjunto de evaluación. En este caso, el texto correcto es 18928. Ambos dígitos del 8 se encuentran parcialmente borrados.

En la Tabla 5.4 se encuentran los resultados de cada uno de los modelos de OCR aplicados a la imagen de la Figura 5.10. En este caso Tesseract detecta correctamente el texto y Calamari no, se obtiene el mismo error en ambos modelos.

#### 5.5.1.2. OCR nombre del índice

El segundo modelo entrenado con datos del índice tiene como objetivo el reconocimiento de nombres. En esta ocasión, se etiquetaron 767 imágenes de nombres

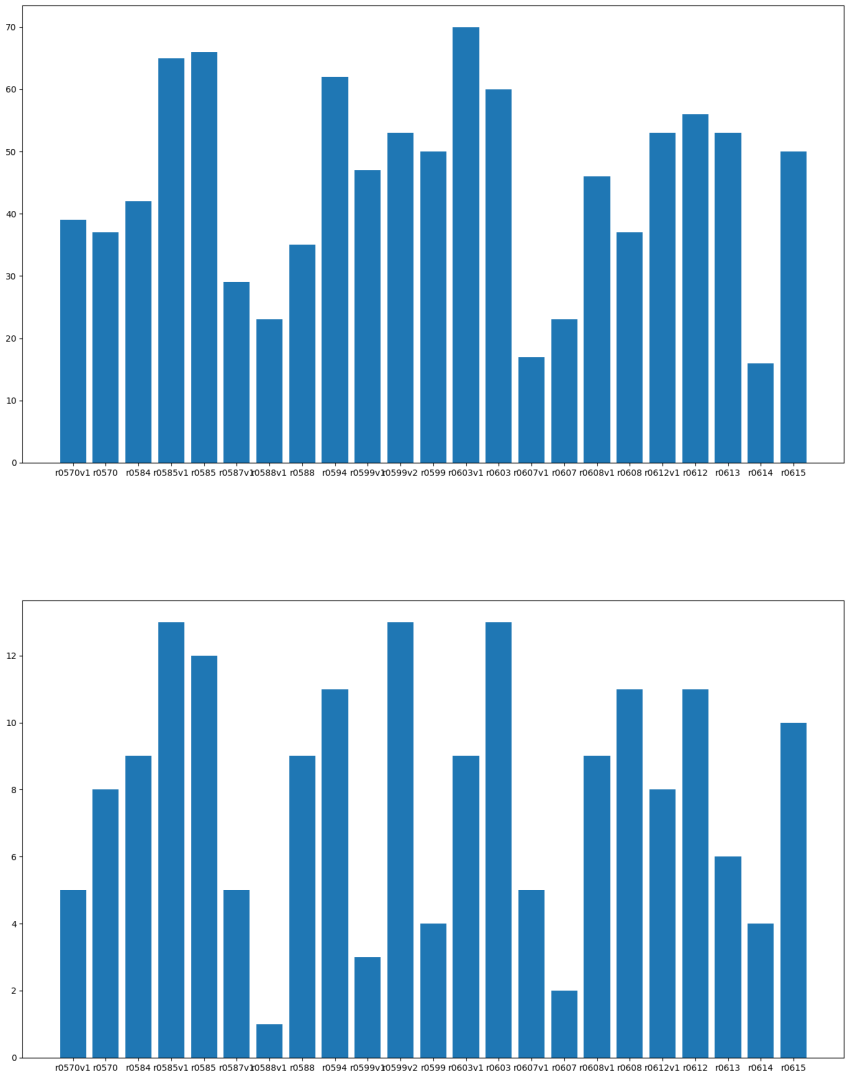



**Figura 5.9:** Histograma de los conjuntos de datos utilizados para entrenar y evaluar el modelo de OCR para reconocer los números de ficha del índice. El gráfico superior es para el conjunto de entrenamiento y el inferior para el conjunto de evaluación. Se observa la cantidad de imágenes tomadas de cada rollo.

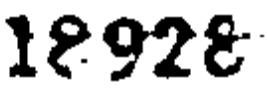


**Figura 5.10:** Ejemplo de número de ficha en una línea del índice.

**Tabla 5.4:** Ejemplo de los resultados del OCR para una imagen con un número de ficha del índice.

| OCR | Resultado |
|---|---|
| Texto correcto | 18928 |
| Calamari baseline | 12928 |
| Calamari ajustado | 12928 |
| Tesseract baseline | 18928 |
| Tesseract ajustado | 18928 |

de las cuales 652 fueron utilizadas en el conjunto de entrenamiento y 115 fueron utilizadas en el conjunto de evaluación. El proceso de etiquetado de estos datos demandó aproximadamente cinco horas.

En la Figura 5.11 se pueden apreciar dos histogramas con la cantidad de nombres etiquetados de cada rollo, uno para el conjunto de entrenamiento y otro para el conjunto de evaluación. En ambos conjuntos se tiene imágenes de todos los rollos. Además, las distribuciones son similares.

Los resultados de entrenar Calamari y Tesseract se encuentran en la Tabla 5.5. Tanto antes como luego de entrenar, Calamari muestra mejores resultados que Tesseract. Con los modelos ajustados, el error es sensiblemente más pequeño con Calamari que con Tesseract, siendo el error del modelo de Tesseract ajustado en el conjunto de evaluación similar al error de Calamari con el modelo de Luisa.

**Tabla 5.5:** Resultados OCR para la columna de nombre en el índice.

| | CER Entrenamiento | CER Evaluación |
|---|---|---|
| Calamari baseline | 0,0766 | 0,0917 |
| Calamari ajustado | 0,0025 | 0,0142 |
| Tesseract baseline | 0,1138 | 0,1268 |
| Tesseract ajustado | 0,0471 | 0,0942 |

Para ambos modelos entrenados con datos del índice se obtienen resultados satisfactorios. No se llevan a cabo pruebas en las otras columnas, dado que la tipografía utilizada es la misma en todas ellas.

En la Figura 5.12 se puede observar un ejemplo perteneciente al conjunto de evaluación.

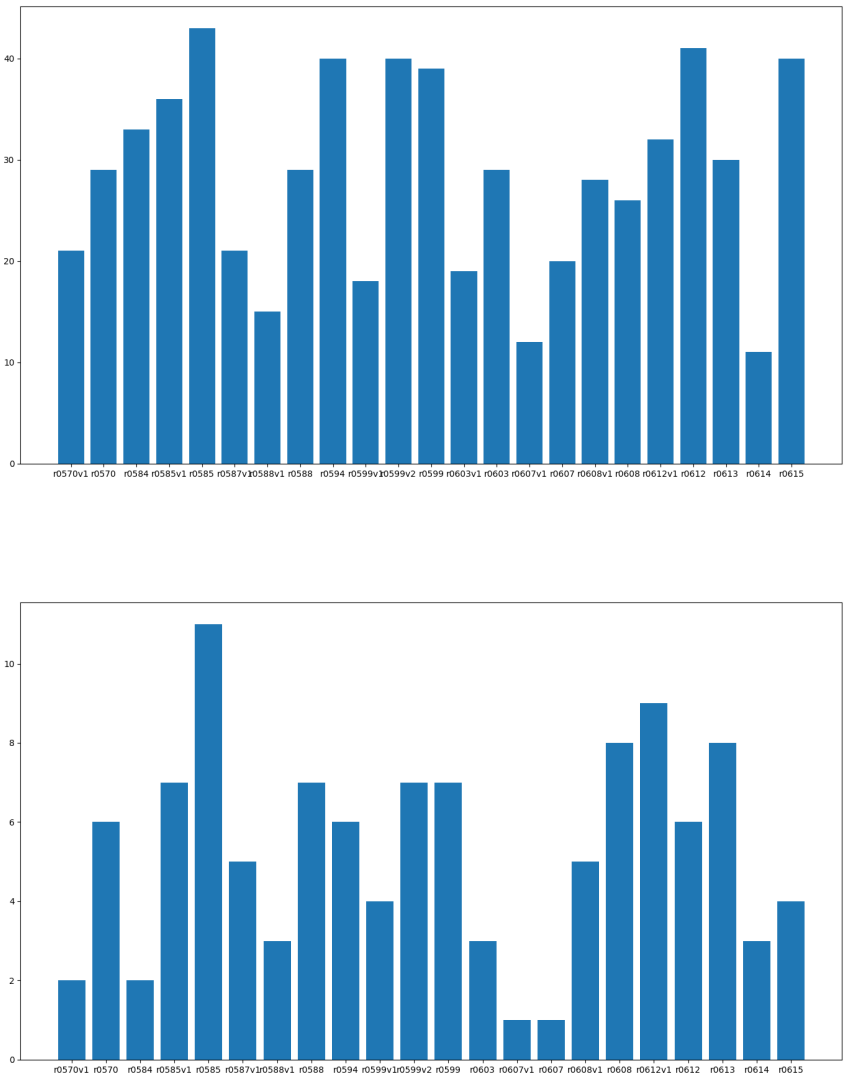



**Figura 5.11:** Histograma de los conjuntos de datos utilizados para entrenar y evaluar el modelo de OCR para reconocer los nombres de las fichas del índice. El gráfico superior es del conjunto de entrenamiento y el inferior del conjunto de evaluación. Se observa la cantidad de imágenes tomadas de cada rollo.

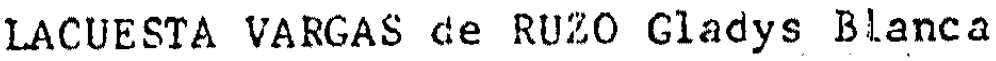



**Figura 5.12:** Ejemplo de nombre en una línea del índice.

En la Tabla 5.6 se encuentran los resultados de cada uno de los modelos de OCR aplicados a la imagen de la Figura 5.12. En este caso Tesseract detectaba correctamente el nombre antes del entrenamiento pero luego de entrenar no lo hace. En el caso de Calamari, el texto es detectado correctamente antes y después de entrenar.

**Tabla 5.6:** Ejemplo de los resultados del OCR para el nombre de un individuo extraído del índice.

| OCR | Resultado |
|---|---|
| Texto correcto | LACUESTA VARGAS de RUZO Gladys Blanca |
| Calamari baseline | LACUESTA VARGAS de RUZO Gladys Blanca |
| Calamari ajustado | LACUESTA VARGAS de RUZO Gladys Blanca |
| Tesseract baseline | LACUESTA VARGAS de RUZO Gladys Blanca |
| Tesseract ajustado | LAGCUESTA VARGAS de RUZO Gladys Blanca |

### 5.5.2. Campos de las fichas

Una vez concluida la clasificación de partes de ficha, para las partes frontales se dispone de imágenes correspondientes a los campos detectados y extraídos. En particular, hay 27.096 imágenes de nombres y 27.096 imágenes de apellidos.

La cantidad de rollos es ahora 14, ya que, a diferencia del índice, no se utilizaron todas las versiones de todos los rollos como fue explicado en el capítulo 4.

#### 5.5.2.1. OCR nombres y apellidos

A diferencia del índice, en este caso no se cuenta con nombres y apellidos todos juntos en una misma imagen, ya que en las fichas están por separado. Siguiendo una

estrategia similar a la utilizada para las columnas del índice, se etiquetan imágenes seleccionadas al azar de todos los rollos, tanto de nombres como de apellidos.

Se construye un conjunto de datos con un total de 5.000 imágenes, donde 2500 son de nombres y 2500 de apellidos. El tiempo total requerido para etiquetar fue de aproximadamente 12 horas. Se genera un conjunto de entrenamiento con 4000 imágenes y otro conjunto de evaluación con 1000 imágenes. Para cada uno de estos conjuntos, la mitad de las imágenes son nombres y la otra mitad son apellidos.

El conjunto de datos es sensiblemente más grande que el utilizado para entrenar los modelos para las columnas del índice debido a que en esta ocasión hay una variedad de tipografías e iluminación mucho mayor.

Un histograma con la cantidad de nombres utilizado en cada rollo se encuentra en la Figura 5.13. La imagen superior es del conjunto de entrenamiento y la inferior del conjunto de evaluación. En ambos conjuntos hay imágenes de todos los rollos y en proporciones similares.

Los resultados del OCR, tanto del modelo base como el modelo entrenado se pueden observar en la Tabla 5.7. El modelo de Calamari entrenado utilizando los datos de Luisa tiene un rendimiento ligeramente superior que el modelo de Tesseract entrenado con los datos de Luisa.

Al entrenar con los datos de nombres y apellidos, se observa una mejora significativa de Calamari, obteniendo resultados muy buenos tanto en el conjunto de entrenamiento como en el conjunto de evaluación, con un error más de tres veces más pequeño en evaluación en comparación con el modelo base entrenado con los datos de Luisa. Tesseract también mejora su rendimiento, obteniendo un error tres veces más pequeño. Sin embargo, sigue siendo más grande que el de Calamari.

**Tabla 5.7:** Resultados OCR para el campo nombres y apellidos de las fichas.

| | CER Entrenamiento | CER Evaluación |
|---|---|---|
| Calamari baseline | 0,2913 | 0,2924 |
| Calamari ajustado | 0,0154 | 0,0794 |
| Tesseract baseline | 0,3365 | 0,3354 |
| Tesseract ajustado | 0,0689 | 0,1145 |

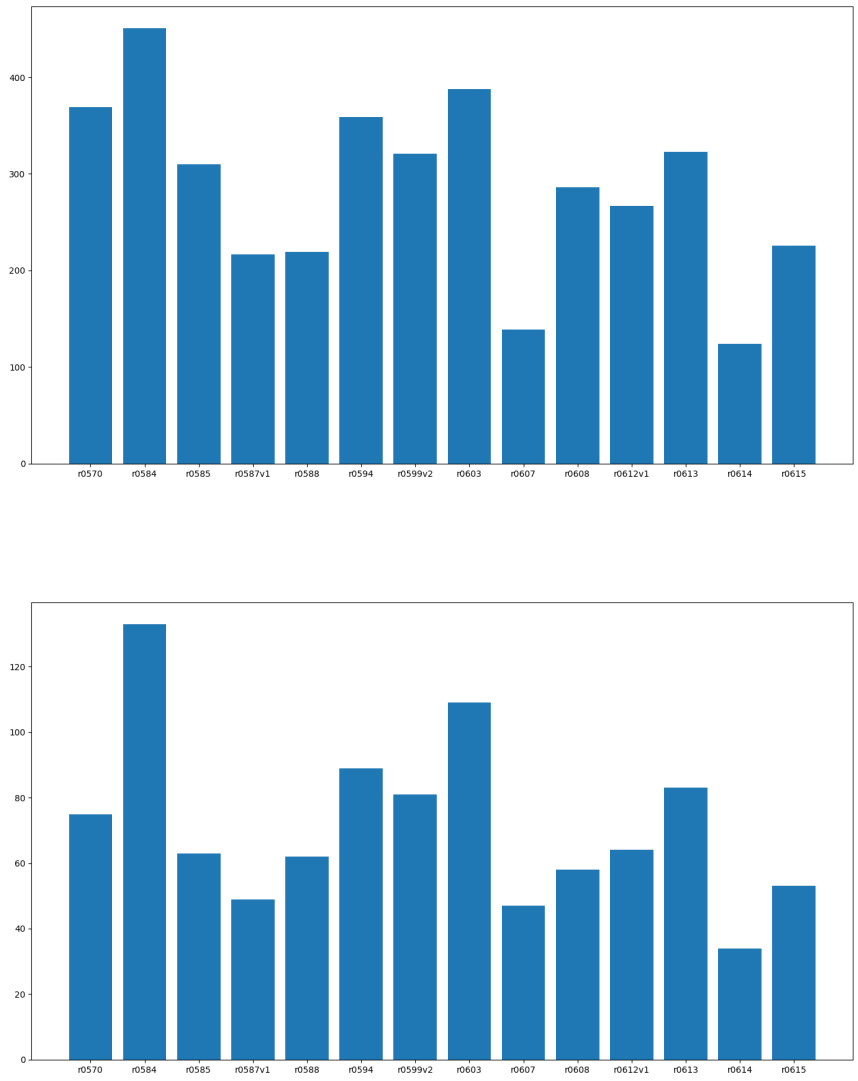



**Figura 5.13:** Histograma de los conjuntos de datos utilizados para entrenar y evaluar el modelo de OCR para reconocer los nombres extraídos de las partes frontales de las fichas. El gráfico superior es del conjunto de entrenamiento y el inferior del conjunto de evaluación. Se observa la cantidad de imágenes tomadas de cada rollo.

Al comparar con los resultados obtenidos para las columnas del índice, el desempeño fue bastante superior. Esto podría atribuirse a que el índice no tiene tantas tipografías diferentes y, en general, hay menos problemas de legibilidad.

En la Figura 5.14 se puede observar un ejemplo perteneciente al conjunto de evaluación. Hay varias letras que se encuentran parcialmente borradas.

**Figura 5.14:** Ejemplo de apellido en una parte de ficha.

En la Tabla 5.8 se encuentran los resultados de cada uno de los modelos de OCR aplicados a la imagen de la Figura 5.14. En este caso, solamente Calamari entrenado logró detectar correctamente el texto. Se observa que Tesseract detecta textos diferentes antes y luego de entrenar, ambos incorrectos.

**Tabla 5.8:** Ejemplo de los resultados del OCR para apellidos extraídos de una parte de ficha.

| OCR | Resultado |
|---|---|
| Texto correcto | KARTESZ WASERZTEJN |
| Calamari baseline | MARTESZASFRETEJ |
| Calamari ajustado | KARTESZ WASERZTEJN |
| Tesseract baseline | KARTESZ MWASERZTEEÑNNN |
| Tesseract ajustado | FRTESZ WASERTX |

#### 5.5.2.2. OCR cédulas

Los modelos de OCR para el reconocimiento de cédulas se entrenaron utilizando imágenes etiquetadas de todos los rollos seleccionadas aleatoriamente. Se construye un conjunto de datos con 3000 imágenes, el tiempo de etiquetado insumió aproximadamente 8 horas. Este conjunto se divide en 2400 imágenes para el conjunto de entrenamiento y 600 imágenes para el conjunto de evaluación.

Un histograma con la cantidad de imágenes de cédulas utilizadas de cada rollo se encuentra en la Figura 5.15. La imagen superior es del conjunto de entrenamiento

y la inferior del conjunto de evaluación. Una vez más, en ambos conjuntos hay imágenes de todos los rollos y en proporciones similares.

Los resultados obtenidos mediante el modelo base entrenado con datos de Luisa y luego de realizado el entrenamiento se encuentran en la Tabla 5.9. Ambos OCR demostraron un buen desempeño luego de finalizado el entrenamiento, mejorando sustancialmente en comparación con el modelo base. Comparando los modelos ajustados, Calamari logra un desempeño levemente superior a Tesseract en el conjunto de evaluación aunque el error de Tesseract es apenas inferior al de Calamari en el conjunto de entrenamiento. En conclusión, el desempeño de ambos es similar con Calamari superando ligeramente a Tesseract.

**Tabla 5.9:** Resultados OCR para el campo cédulas.

| | CER Entrenamiento | CER Evaluación |
|---|---|---|
| Calamari baseline | 0,2051 | 0,1848 |
| Calamari ajustado | 0,0044 | 0,0157 |
| Tesseract baseline | 0,1691 | 0,1563 |
| Tesseract ajustado | 0,0035 | 0,0187 |

En la Figura 5.16 se puede observar un ejemplo de una imagen perteneciente al conjunto de evaluación. Hay un dígito uno que se encuentra parcialmente borrado.

En la Tabla 5.10 se encuentran los resultados de cada uno de los modelos de OCR aplicados a la imagen de la Figura 5.16. Tesseract entrenado logra detectar correctamente los dígitos. Calamari falla tanto con el modelo base como con el modelo entrenado. Sin embargo, se observa una mejoría ya que con el modelo base ni siquiera detectaba números y con el modelo ajustado si lo hace.

**Tabla 5.10:** Ejemplo de los resultados del OCR para una imagen de una cédula.

| OCR | Resultado |
|---|---|
| Texto correcto | 1.180.102 |
| Calamari baseline | doE |
| Calamari ajustado | 1.80.165 |
| Tesseract baseline | 1. 80 10 |
| Tesseract ajustado | 1.180.102 |

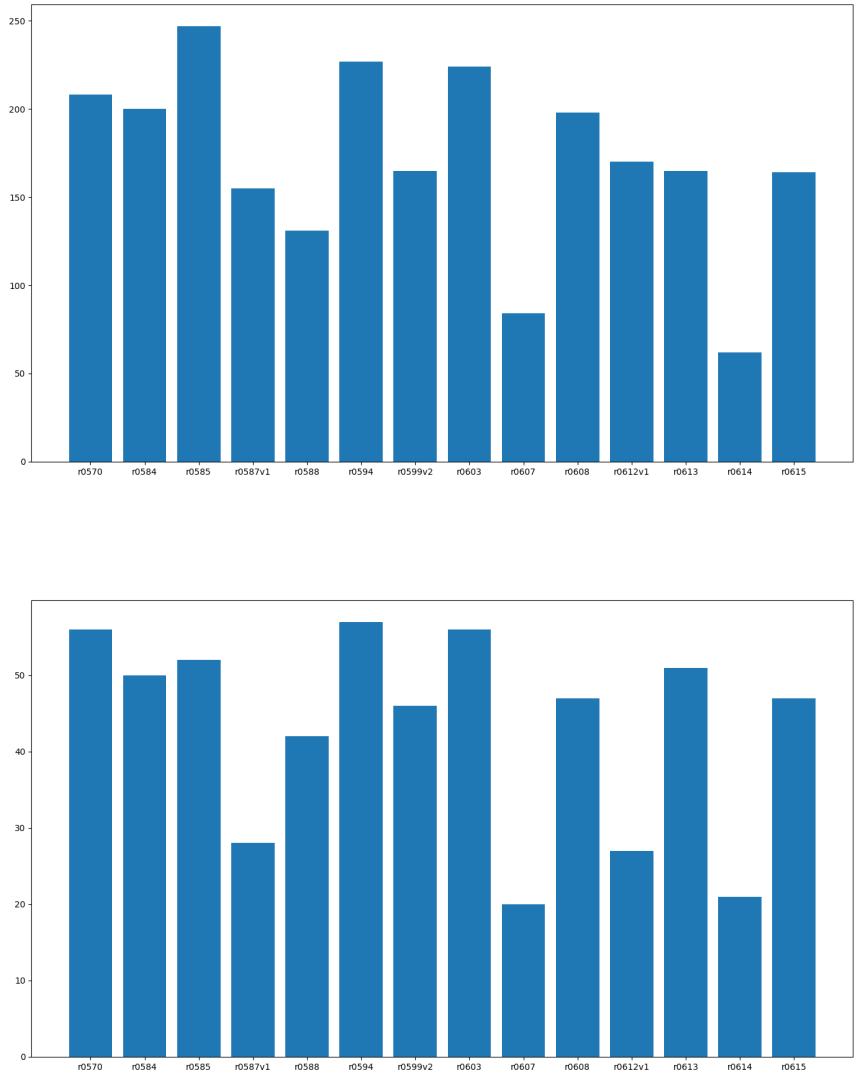



**Figura 5.15:** Histograma de los conjuntos de datos utilizados para entrenar y evaluar el modelo de OCR para reconocer imágenes de cédulas extraídas de las partes frontales de las fichas. El gráfico superior es del conjunto de entrenamiento y el inferior del conjunto de evaluación. Se observa la cantidad de imágenes tomadas de cada rollo.

**Figura 5.16:** Ejemplo de una cédula extraída de una parte de ficha.

### 5.5.3. Número de ficha

En esta sección, se presentan los resultados obtenidos de entrenar un modelo de OCR para los números de ficha extraídos de las partes de ficha. Este número es de gran importancia ya que permite asociar todas las partes de ficha y la línea del índice pertenecientes a la misma persona.

Este número se obtiene del índice mediante la separación de las líneas y columnas, con el algoritmo desarrollado en 4.4. Asimismo, el número se extrae de las partes de ficha mediante el algoritmo presentado en 4.3.

Con el fin de generar un conjunto de datos para entrenar y evaluar los modelos de OCR, se seleccionan nuevamente imágenes de los números extraídos de manera aleatoria de todos los rollos. El total de números extraídos es de 61.294. Tras un proceso de etiquetado que demandó aproximadamente 6 horas se consigue etiquetar 1.863 imágenes. De estas, se utilizaron 1.583 para conformar el conjunto de entrenamiento, mientras que las restantes 280 se destinaron al conjunto de evaluación.

En la Figura 5.17 se encuentra el histograma que enseña cuantas imágenes hay de cada rollo en cada uno de los conjuntos. La imagen superior corresponde al conjunto de entrenamiento y la inferior al conjunto de evaluación. En ambos conjuntos hay imágenes de todos los rollos.

Los resultados obtenidos con los modelos base y luego de realizado el entrenamiento se encuentran en la Tabla 5.11. Este conjunto fue el que obtuvo peores resultados en comparación a todos los otros considerando los modelos base. En estas imágenes, el ruido de fondo es en muchos casos texto, por lo que una posibilidad es que intenten reconocer ese texto en lugar de los dígitos.

Afortunadamente, los resultados de los modelos entrenados tienen una mejoría notable en comparación con los modelos base. En ambos casos el error se reduce más de diez veces en el conjunto de evaluación. En esta ocasión, los dos errores en el conjunto de evaluación luego de entrenar son casi iguales, siendo el de Tesseract ligeramente más pequeño.

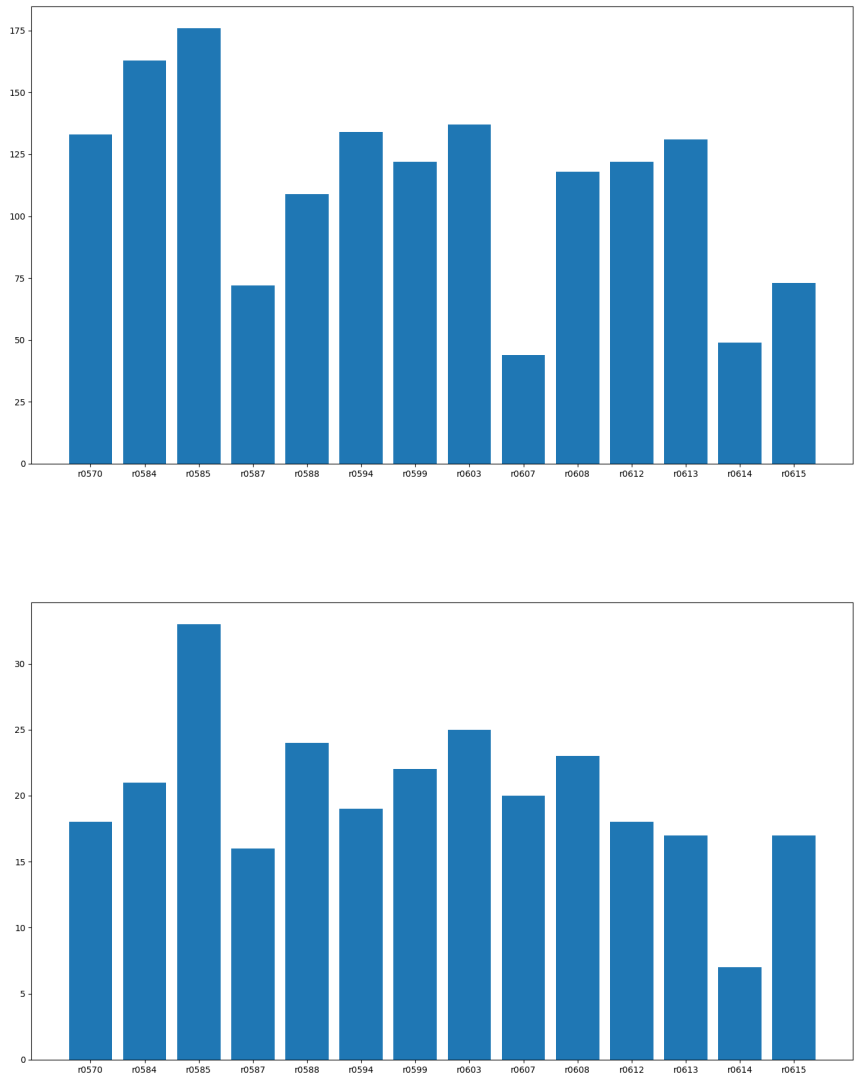



**Figura 5.17:** Histograma de los conjuntos de datos utilizados para entrenar y evaluar el modelo de OCR para reconocer los números de ficha extraídos. El gráfico superior es del conjunto de entrenamiento y el inferior del conjunto de evaluación. Se observa la cantidad de imágenes tomadas de cada rollo.

Con estos resultados, se espera que ambos modelos tengan buen desempeño al aplicar el OCR a todas las imágenes.

**Tabla 5.11:** Resultados OCR para el número de ficha extraído.

| | CER Entrenamiento | CER Evaluación |
|---|---|---|
| Calamari baseline | 0,4869 | 0,4570 |
| Calamari ajustado | 0,0293 | 0,0333 |
| Tesseract baseline | 0,5012 | 0,4913 |
| Tesseract ajustado | 0,0139 | 0,0301 |

En la Figura 5.18 se puede observar un ejemplo perteneciente al conjunto de evaluación.

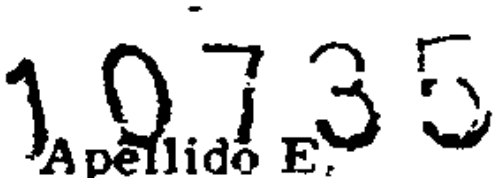



**Figura 5.18:** Ejemplo del número de ficha extraído de una parte de ficha.

En la Tabla 5.12 se encuentran los resultados de cada uno de los modelos de OCR aplicados a la imagen de la Figura 5.18. Tesseract entrenado logra detectar correctamente el número. Calamari falla tanto con el modelo base como con el modelo entrenado, mostrando el mismo error.

**Tabla 5.12:** Ejemplo de los resultados del OCR para el número de ficha extraído de una parte de ficha.

| OCR | Resultado |
|---|---|
| Texto correcto | 10735 |
| Calamari baseline | 10135 |
| Calamari ajustado | 10135 |
| Tesseract baseline | 0 135 |
| Tesseract ajustado | 10735 |

### 5.5.4. Comparación con OCR comercial

Si bien se obtuvieron buenos resultados al entrenar Calamari y Tesseract, surge la interrogante acerca de la calidad de estos resultados en comparación con un OCR

comercial. La dificultad radica en que los OCR comerciales más destacados se encuentran en la nube y por lo tanto su uso es muy limitado con los datos extraídos de las fichas o del índice.

Sin embargo, el número de ficha es un número que va aumentando secuencialmente. En consecuencia, este dato por sí solo no representa ningún riesgo en subirlo a la nube. Por consiguiente, se opta por llevar a cabo una comparación del OCR entrenado para el número de ficha de la columna del índice. Se emplean los mismos conjuntos de datos que se utilizaron para entrenar Calamari y Tesseract, con el fin de garantizar que las imágenes subidas no contienen por error parte del nombre.

El OCR comercial elegido es Document.AI «Document AI», s.f. de Google. Es una solución en la nube para el procesamiento de documentos. Brinda capacidades de análisis, búsqueda y extracción de datos. En particular, incluye modelos de OCR entrenados en grandes cantidades de datos. Además, cuenta con una biblioteca de Python que permite ejecutar el OCR de manera sencilla.

En la Tabla 5.13 se encuentran nuevamente los resultados de los OCR entrenados para detectar el número de ficha de la columna en el índice, agregando también los resultados obtenidos por Document.AI.

Se puede observar que Document.AI presenta un desempeño significativamente inferior tanto en comparación con modelo de Tesseract ajustado como con el modelo de Calamari ajustado. Si se consideran los modelos base entrenados con los datos de Luisa, Tanto Calamari como Tesseract siguen siendo superiores a Document.AI.

**Tabla 5.13:** Resultados OCR para el número de ficha en el índice, agregando los resultados de Document.AI.

| | CER Entrenamiento | CER Evaluación |
|---|---|---|
| Calamari baseline | 0,0392 | 0,0385 |
| Calamari ajustado | 0,0011 | 0,0033 |
| Tesseract baseline | 0,0656 | 0,0549 |
| Tesseract ajustado | 0,00048 | 0,0033 |
| Document.AI | 0,1498 | 0,1703 |

Se puede concluir que Document.AI tuvo un desempeño inferior al logrado con Tesseract y Calamari en todos los casos. No obstante, es relevante destacar que Document.AI es una herramienta considerablemente más completa que un OCR, ya

que incorpora etapas de preprocesamiento que permiten, por ejemplo, detectar las líneas de texto. Esta es una funcionalidad que Calamari no incluye pero sí lo hace Tesseract.

Se espera que para las columnas restantes del índice y los campos de las fichas los resultados sean similares, dado que los datos utilizados con Document.AI fueron los que presentaban menos dificultades y son los más uniformes.

### 5.5.5. Reconocimiento de cédulas utilizando template matching

En el capítulo 4, se expuso previamente que se realizó un intento de reconocimiento de cédulas mediante el uso de *template matching* para reconocimiento de cédulas. El método fue finalmente descartado debido a bajo rendimiento.

Con el fin de obtener los patrones, se trabajó con el rollo 570 de manera iterativa. Se tomó un conjunto de imágenes de cédulas etiquetado, se generó un patrón para cada dígito del cero al nueve, un patrón para el punto y uno para el guion.

En ese momento, se lleva a cabo la técnica de reconocimiento para todas las imágenes del conjunto y se observan los errores. Para las cédulas reconocidas erróneamente, se generan patrones utilizando la imagen de dicha cédula de los dígitos reconocidos incorrectamente. A continuación, se procede a repetir el proceso de reconocimiento, utilizando la nueva cantidad de patrones.

Después de repetir este proceso en varias ocasiones, se obtuvo una cantidad considerablemente elevada de patrones, sin embargo, el error no disminuía. Empezó a ocurrir que, en ocasiones, se añadían patrones nuevos y debido al ruido, dígitos que antes eran reconocidos de manera correcta ahora no lo eran. Además, la velocidad de ejecución cada vez se volvía más lenta.

La cantidad final de patrones es de 153. En la Tabla 5.14 se puede observar la cantidad de patrones extraídos para cada dígito. Los caracteres que presentaron más problemas de reconocimiento son aquellos que cuentan con la mayor cantidad de patrones. Por ejemplo, el número ocho suele ser confundido con el cero o el tres.

Además, se presentaron casos en los que ninguno de los dígitos alcanzaba el umbral de similitud requerido, por lo tanto, era necesario añadir un nuevo patrón.

**Tabla 5.14:** Cantidad de patrones generados para cada carácter

| Carácter | Cantidad de patrones |
|---|---|
| Dígito 0 | 12 |
| Dígito 1 | 14 |
| Dígito 2 | 10 |
| Dígito 3 | 12 |
| Dígito 4 | 16 |
| Dígito 5 | 16 |
| Dígito 6 | 14 |
| Dígito 7 | 9 |
| Dígito 8 | 27 |
| Dígito 9 | 15 |
| Dígito punto | 5 |
| Dígito guion | 2 |

Para realizar una comparación con Tesseract y Calamari, se utiliza *template matching* en el mismo conjunto de datos que se utilizó para evaluar dichos OCR. Es posible que alguna de las imágenes en este conjunto también haya sido utilizada para generar patrones. El CER obtenido es de 0,1651. Este resultado es mejor que el obtenido Tesseract utilizando el modelo base y apenas superior al CER obtenido por Calamari en el modelo base. Se considera un desempeño muy bajo si se compara con los modelos entrenados específicamente para reconocer cédulas de ambos OCR. Los resultados de Tesseract y Calamari con las cédulas se encuentran detallados en la Tabla 5.9.

### 5.5.6. Corrección de resultados

#### 5.5.6.1. Columnas Organización 1 y Organización 2

Cómo fue mencionado en la Sección 4.5.3, para las columnas Organización 1 y Organización 2 se intentó corregir el resultado del OCR utilizando un diccionario que fue generado manualmente leyendo las hojas del índice.

Para el caso de las organizaciones, las palabras reconocidas que se encontraban cerca en distancia de edición con una palabra del diccionario (menor o igual a 1) eran reemplazadas por esa palabra. Los resultados se analizan por separado para Tesseract y Calamari, comenzando por Calamari.

En la Figura 5.19 se puede observar el resultado obtenido con Calamari en las columnas de organización del índice agrupados por organización, junto con los resultados obtenidos luego de realizar la corrección. Las gráficas de la izquierda corresponden a la columna Organización 1 y las gráficas de la derecha a la columna Organización 2. Las gráficas superiores corresponden a los resultados sin corregir y las gráficas inferiores a los resultados corregidos.

Se incluye una columna denominada 'Otro' que representa los casos en los que el OCR no detectó una organización perteneciente al diccionario. Se excluyeron las celdas en las que el OCR no detectó ningún texto.

En ambos casos, luego de realizar la corrección, se observa una reducción de la cantidad de elementos en la columna 'Otro' y aumento en las otras columnas, algo que indicaría que se logró corregir el resultado en muchos casos.

Se analizan algunos casos y se encuentran tanto correcciones acertadas, como por ejemplo: 'UJG' corregido a 'UJC', 'PYP' corregido a 'PVP' así como correcciones incorrectas. Por ejemplo, 'PCE' fue corregido a 'PC' pero lo correcto era 'PCR'. o 'JC' fue corregido a 'PC' cuando lo correcto era 'UJC'. Estas situaciones se producen debido a que la distancia es la misma tanto al valor correcto como a otros valores del diccionario.

Los resultados para Tesseract se encuentran disponibles en la Figura 5.20, tanto antes como luego de corregir. Nuevamente, hay una columna 'Otro' que contiene los resultados que no pertenecen al diccionario agrupados. Con este OCR hay muchos más resultados en la columna 'Otro' que en el caso de Calamari, lo que demuestra la superioridad de Calamari sobre Tesseract en estos datos.

En esta oportunidad, si bien hay muchos resultados corregidos, alrededor de 6000 para Organización 1 y aproximadamente 900 en Organización 2, la columna 'Otro' sigue siendo la que contiene más ejemplos en el caso de Organización 2

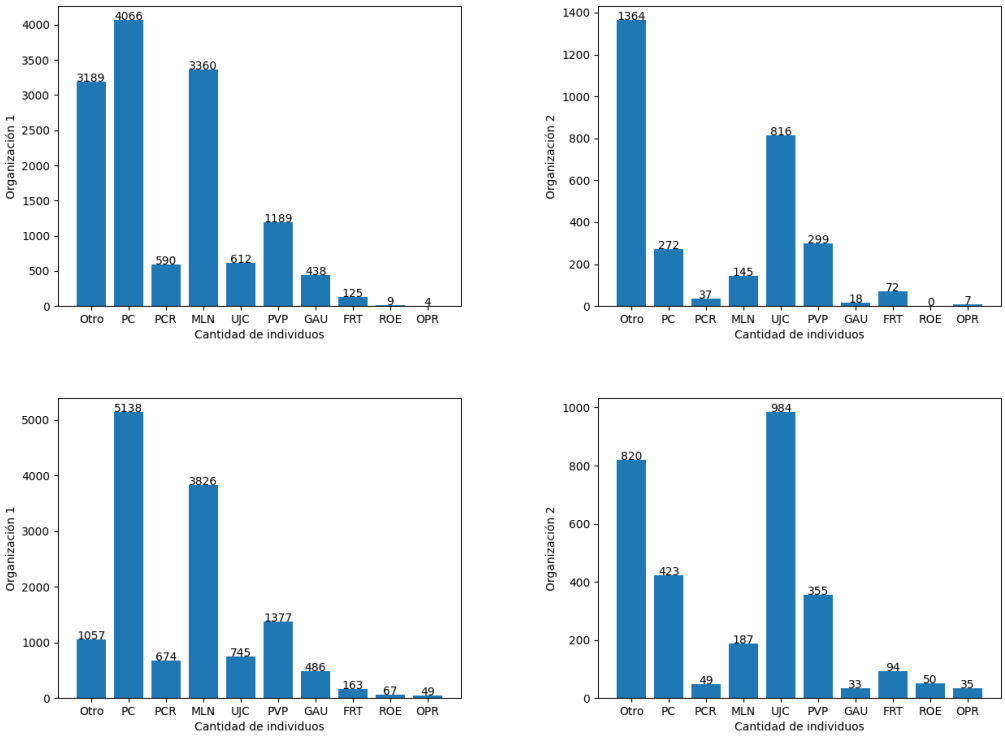



**Figura 5.19:** Resultados de Calamari en las columnas de organización del índice, se muestra la cantidad de individuos en cada organización. La columna Otro representa casos en los que el resultado del OCR no pertenece al diccionario. La esquina superior izquierda representa la columna Organización 1 antes de la corrección. La esquina inferior izquierda representa la columna Organización 1 luego de corregidos los resultados. De manera similar, la esquina superior derecha contiene los resultados de Organización 2 antes de corregir y la superior derecha los resultados de Organización 2 luego de corregir.

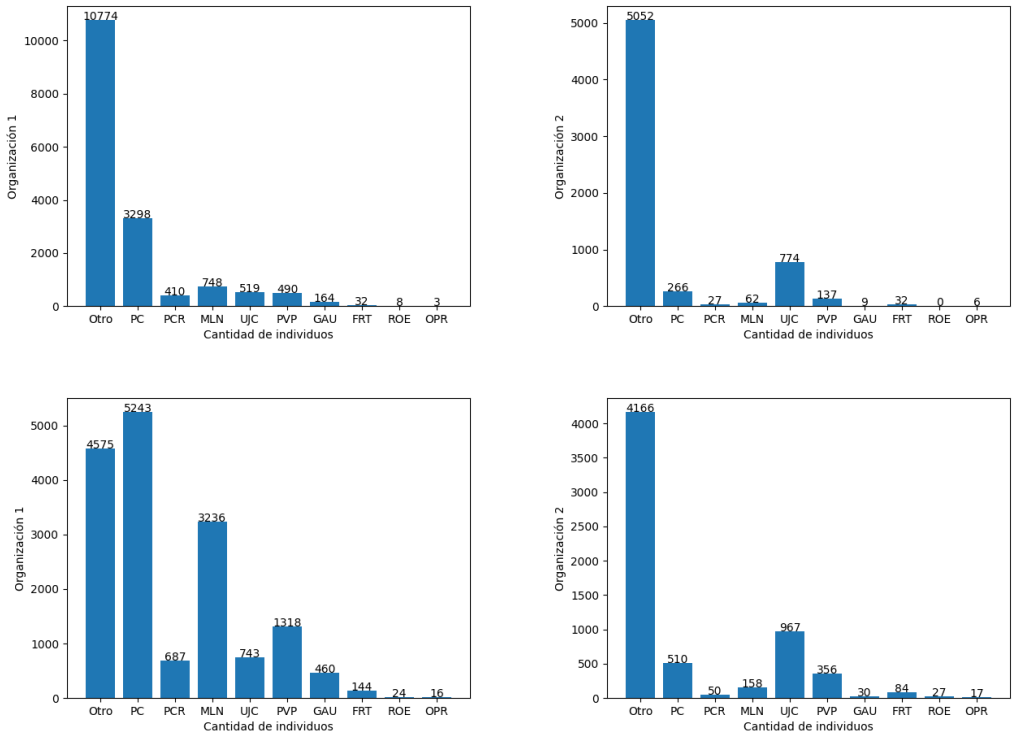



**Figura 5.20:** Resultados de Tesseract en las columnas de organización del índice. Se muestra la cantidad de ejemplos detectados en cada organización. La columna Otro representa casos en los que el resultado del OCR no pertenece al diccionario. La esquina superior izquierda representa la columna Organización 1 antes de la corrección. La esquina inferior izquierda representa la columna Organización 1 luego de corregidos los resultados. De manera similar, la esquina superior derecha contiene los resultados de Organización 2 antes de corregir y la superior derecha los resultados de Organización 2 luego de corregir.

luego de realizada la corrección.

Para llevar a cabo una evaluación objetiva, se seleccionan aleatoriamente cinco hojas del índice y se analizan los resultado en las columnas de organización antes y después de realizar la corrección. En total, entre las dos columnas de las hojas seleccionadas, se cuenta con 99 ejemplos. Antes de realizar la corrección, con Calamari se encontraron 89 ejemplos correctos, mientras que con Tesseract solamente se obtuvieron 54 ejemplos correctos. Después de realizar la corrección, el número de ejemplos correctos aumentó a 95 con Calamari y a 84 con Tesseract. Se observa una mejora en ambos casos, siendo especialmente notable en el caso de Tesseract.

Se puede concluir que, en el caso de las columnas de organización, la corrección con diccionario resultó efectiva para subsanar errores de los OCR. Sin embargo, es posible introducir errores en caso de que lo detectado por el OCR se encuentre a la misma distancia de más de una palabra del diccionario.

#### 5.5.6.2. Apellidos en el índice

El segundo lugar donde se aplicó corrección con diccionario fue para los apellidos de la columna del índice que contenía los nombres.

De un total de 61.758 líneas, en 22.903 hubo correcciones de Calamari en apellidos. Esta cifra es bastante elevada dado el buen rendimiento del OCR al momento de evaluar el modelo entrenado. Al analizar ejemplos manualmente, se detecta que hay muchos apellidos que fueron correctamente detectados pero no pertenecían al diccionario. Como consecuencia, esos apellidos fueron corregidos al más cercano en distancia de edición que en muchas ocasiones se encontraba a una distancia muy grande. Con Tesseract, se observan resultados similares.

El tener un diccionario de mala calidad o incompleto, como en este caso, degrada el rendimiento del sistema en lugar de mejorarlo.

### 5.5.7. Conclusiones

Se llevó a cabo un entrenamiento y evaluación de dos modelos diferentes de OCR para varios campos de las fichas y columnas del índice. En todos los casos se obtuvieron resultados razonables por lo que se espera buenos resultados en el reconocimiento del texto.

Se exploró una técnica alternativa el OCR para reconocer caracteres en las cédulas pero fue descartada debido a su bajo rendimiento.

Además, se realizó una breve comparación de desempeño con un OCR comercial de Google, en la cual tanto Calamari como Tesseract entrenado demostraron su superioridad. Esto evidencia que el etiquetar datos y entrenar un modelo puede llegar a brindar más valor que utilizar el mejor OCR de la actualidad pero entrenado utilizando los datos de otro problema.

Al comparar los dos OCR utilizados, se puede concluir que Calamari muestra ser superior a Tesseract en más casos. En particular, Calamari es superior a Tesseract en los modelos para reconocer nombres del índice, nombres y apellidos de las fichas y en el modelo de reconocimiento de cédulas. Por otro lado, Tesseract es superior a Calamari únicamente en el modelo de reconocimiento de números de ficha (de las fichas). Finalmente, ambos OCR tienen el mismo desempeño al reconocer números de ficha del índice.

En última instancia, es importante resaltar que el entrenamiento requiere contar con datos debidamente etiquetados o tiempo disponible para etiquetarlos. El etiquetado es un proceso muy costoso en tiempo. No obstante, a la luz de los resultados obtenidos, en el presente caso se puede concluir que fue una decisión acertada.

## 5.6. Evaluación final

Hasta el momento, se evaluó cada etapa del proceso individualmente, pero surge la interrogante de qué resultado se obtienen al evaluar el sistema en su totalidad. Con

este fin, se seleccionan tres partes frontales de ficha aleatoriamente y se analizan los resultados obtenidos por los OCR para el nombre, la cédula y el número de ficha. Luego, esa misma parte de ficha se procesa con Tesseract utilizando el modelo de español incluido por defecto y el modo de detección automático que detecta líneas de texto, pasándole como entrada la imagen con la parte de ficha entera.

Para la primera parte de ficha evaluada, la solución implementada logra detectar correctamente el número de cédula con Calamari, el apellido con Tesseract y el nombre detectado fue LENARDO, mientras que el nombre correcto es LEONARDO (se encuentra a uno en distancia de edición). Con ambos métodos el número de ficha extraído fue 344 y el correcto era 5344.

Al ejecutar Tesseract sobre la parte de ficha entera, se logran detectar las etiquetas de los campos pero el texto de estos en general no aparece o aparecen otros caracteres ilegibles. En el listado 5.1 se puede observar este resultado. El número de ficha es extraído correctamente, 5344 y aparece algo cercano al apellido, CAGGTANT cuando lo correcto es CAGGIANI. No se extrae ni el nombre ni la cédula.

```
    ESTARE oo rm O  –
| 5344 3 A
Apellido P. _. Apellido M. Apellido F. | ler. Nombre 2do. Nombre D) Fechas 4 Ver
CAGGTANT Írrew RIO – ha 2 Accept
ON A LR TE A CO. ee
Nacionalidad: nn A Fecha Nac. / Eladio cis
Meg. Potooooooccccccoccnnccenco nano rcnnn cnn Indi Dact.coooocccicccccccccca icono cnc ¡ÓN
Cadell oocccicccccconononcoccnanannnnnnnos OJO ooccoccen nana n coco nnnonncnnnnncennnnns Cajas
      occccocococononnnonnononanannnnnoos Nariz:

Fecha de requerido: ccoo ec MO

AT ATA RC
```

**Listing 5.1:** Resultado Tesseract sobre toda la primera parte de ficha usada en la evaluación final.

En el caso de la segunda parte de ficha evaluada, el método desarrollado logra detectar correctamente el número de ficha y el apellido. El nombre lo detecta incorrectamente, Calamari detecta ALRERT y Tesseract ALBERT cuando lo correcto es ALBERTO. Esta parte de ficha no tiene la cédula.

El resultado de Tesseract se encuentra disponible en el listado 5.2. Nuevamente se detectan las etiquetas de los campos correctamente en muchos casos. Además, algunos campos se detectan correctamente. Por ejemplo, el nombre o la edad. Sin embargo, no se detecta el número de ficha.

```
Apellido P. Apellido M. Apellido E.
PATDULIT " 21

Ter. Nombre 2do. Nombre

ALBERTO .-

| "ALIAS": Ct: e C.C. Serie: N.o
| Nacionalidad: C riental.-Est Civil: CuSAdO +- Fecha Nac/Edad; 48 AÑOS. tugar
Peg, Fat. A Estat Peso
Gabe O A cacon

a A A AA AA A

Domicilio La FÉ 706 Anto. 2. ss entre/casi
- Ocupación : a Dirección trabajo a

_ Nombre esposa/concubina

Nombre hijos

Ver fi cha

fa tana?

enteg

a

der Hem, de te D ANTI

PP PPP" EN ARTTDA4> OA +7
```

**Listing 5.2:** Resultado Tesseract sobre toda la segunda parte de ficha usada en la evaluación final.

Finalmente, para la última parte de ficha evaluada, el sistema logra detectar correctamente el número de ficha, nombre, apellidos y cédula. El resultado de aplicar Tesseract sobre toda la parte de ficha se encuentra en el listado 5.3. Se detectan en muchos casos las etiquetas de los campos. En algunos casos, también el contenido. Por ejemplo, la cédula de identidad fue detectada correctamente, el número de ficha (29562) también. Los apellidos son URIETA RIVERO y Tesseract logró detectar ORIETA RIVERO. El nombre no se logra detectar.

```
13
Apellido P. Apeñiido M. Apeilido E. tor. Nombre 266. Nombre [  Fasse

|
ORIETA RIVERO 29562 | size | +72 * |
| >:

ALIAS": c+: 1.454.378 <o: Momtevor__ cc. serio: No

| Nacionalidad: Est. Civil: fecha Nac./Edad: 15 años" Lugar
>>>; UT ----- :
¡Reg. Fot: _. indiv. Dact.: Estat: Peso,

Cabello: Ojos:

¡Otras señas:
! Domicilio: Dgo. TOREES 4261.= _entre/casi:

Ocupación: Dirección trabajo:

a
¡Nombre esposa/concubina:

| Nombre hijos:___ o E e o 0 1672

A A A A A e TA

A A A A A A A A A A A A nn

AAA AS
```

**Listing 5.3:** Resultado Tesseract sobre toda la tercer parte de ficha usada en la evaluación final.

Se puede concluir que el sistema desarrollado tiene un mejor desempeño que Tesseract. Además, al realizar la evaluación hubo algunos detalles que se pasaron por alto. En los casos en los que Tesseract logra detectar correctamente el valor de los campos, no es trivial extraer esa información automáticamente. Por ejemplo, para la última parte de ficha evaluada, resulta muy difícil discernir qué parte del texto detectado corresponde al número y cuál a los apellidos, dado que no existe ningún elemento cercano que pueda utilizarse como referencia. En cuanto a la cédula, si bien se detecta correctamente el número, la etiqueta del campo no lo hace, lo cual dificulta también la tarea de encontrar la cédula en el texto. Otro aspecto que se omitió es que se utilizó Tesseract sobre la parte de ficha ya recortada. En caso de prescindir del sistema desarrollado, sería necesario utilizar la hoja completa, lo que agregaría la dificultad de identificar claramente los límites de cada una de las partes de ficha.

# Capítulo 6

# Conclusiones

En esta investigación se trabajó con el fichero general de la O.C.O.A. perteneciente al *Archivo Berrutti*. El objetivo principal consistió en extraer la mayor cantidad de información posible de manera de poder realizar búsquedas dentro del fichero fácilmente y poder conocer que personas están en el fichero.

Se comenzó por desarrollar una técnica para poder separar todas las fichas pertenecientes a una hoja y almacenarlas cada una en su propio archivo. Luego, se generó una técnica para poder clasificar cada una de las partes de ficha y extraer los campos con información relevante utilizando *template matching*.

Asimismo, se ha trabajado en el procesamiento del índice, el cual se presenta en formato de tabla junto con las fichas. Se ha diseñado una técnica para separar las filas y columnas de la tabla, y se han guardado cada una de las celdas en imágenes individuales.

Para reconocer el texto de las imágenes extraídas se trabajó con dos OCR, Tesseract y Calamari. Ambos han demostrado rendimiento muy bueno en otras investigaciones. Fue necesario dedicar tiempo a etiquetar datos ya que los modelos con los que se contaba no dieron buenos resultados. Luego del entrenamiento de los OCR, se obtuvieron buenos resultados de detección de texto en los campos que se utilizaron para entrenar.

Al realizar una comparación de los resultados obtenidos con un OCR comercial (Document.AI) y con el modelo de español que trae Tesseract por defecto se observa que los resultados obtenidos son muy superiores en ambos casos.

Por último, se ha desarrollado una interfaz web que permite llevar a cabo búsquedas y visualizar los resultados de manera cómoda y eficiente.

# Capítulo 7

# Trabajo Futuro

El fichero general de la O.C.O.A. con el que se trabajó durante esta investigación es uno de los grandes ficheros pertenecientes al *Archivo Berrutti*. Existen otros ficheros mucho más grandes como por ejemplo el fichero general del S.I.D. (Servicio de Información de Defensa) que cuenta con aproximadamente 174.300 fichas, así como otros rollos con fichas de otros organismos como la D.G.I.D. (Dirección General de Inteligencia de Defensa) o ficheros incautados por organismos represores como el fichero del P.C.U (Partido Comunista) o el fichero de la UJC (Unión de Juventudes Comunistas).

Una primer aproximación a extraer información de esos rollos puede consistir en aplicar las mismas técnicas que se utilizaron en este trabajo a esos ficheros. En particular, se puede probar utilizar *template matching* como método de clasificación y extracción de campos para luego intentar entrenar un modelo de Tesseract o Calamari para aplicar OCR a los campos extraídos.

La técnica utilizada para separar las fichas pertenecientes a una misma hoja es posiblemente la que más tenga que modificarse ya que es muy específica a los datos del fichero de la O.C.O.A.

No obstante, el método de detección de texto rotado y alineación así como el método utilizado para separar las líneas del índice son generales y por lo tanto

pueden aplicarse en cualquier otra imagen del *Archivo Berrutti*, no solo en fichas.

En cuanto a las fichas de la O.C.O.A., se debe analizar y profundizar en los resultados obtenidos por los OCR para los campos en los que no se entrenó un modelo. También es posible idear algún sistema de corrección de errores aprovechando que las fichas se encuentran ordenadas tanto alfabéticamente como por el número de ficha.

Sobre la parte de atrás de las fichas con los antecedentes de los individuos no se realizó ningún procesamiento. En el futuro, podría realizarse la correspondiente segmentación de líneas de las mismas y buscar entrenar un modelo de OCR. En este caso, se agrega la dificultad de que hay texto escrito a mano.

# Referencias bibliográficas


Al-Khatatneh, A., Pitchay, S. A., y Al-qudah, M. (2015). A Review of Skew Detection Techniques for Document. *2015 17th UKSim-AMSS International Conference on Modelling and Simulation (UKSim)*, 316-321.

Arica, N., y Yarman-Vural, F. (2002). Optical character recognition for cursive handwriting [Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence]. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(6), 801-813.

Breuel, T. M., Ul-Hasan, A., Al-Azawi, M. A., y Shafait, F. (2013). High-Performance OCR for Printed English and Fraktur Using LSTM Networks [ISSN: 2379-2140]. *2013 12th International Conference on Document Analysis and Recognition*, 683-687.

Burger, W., y Burge, M. J. (2016). *Digital image processing: An algorithmic introduction using java*. Springer.

*Cruzar – Archivos del pasado reciente*. (s.f.). Consultado el 5 de julio de 2023, desde https://cruzar.edu.uy/

Dhakal, P., Munikar, M., y Dahal, B. (2019). One-Shot Template Matching for Automatic Document Data Capture. *2019 Artificial Intelligence for Transforming Business and Society (AITB)*, *1*, 1-6.

*Document AI* [Google Cloud]. (s.f.). Consultado el 5 de julio de 2023, desde https://cloud.google.com/document-ai?hl=es-419

Doermann, D. S., y Tombre, K. (Eds.). (2014). *Handbook of Document Image Processing and Recognition*. Springer.

Etcheverry, L., Agorio, L., Bacigalupe, V., Barreiro, S., Bing, E., Blixen, S., Calegari, D., Cardozo, L., Carpani, F., Chavat, F., Garat, D., Gómez, A., Hernández, F., Marabotto, V., Moncecchi, G., Ramírez, I., Rosá, A., Tiscornia, J., Wonsever, D., … Laguna, R. (2021). A computational framework for the

analysis of the Uruguayan dictatorship archives. En A. Paschke, G. Rehm, J. A. Qundus, C. Neudecker y L. Pintscher (Eds.), *Proceedings of the Conference on Digital Curation Technologies (Qurator 2021), Berlin, Germany, February 8th - to - 12th, 2021* (Vol. 2836). CEUR-WS.org.

*Extracción inteligente de texto y datos con OCR - Amazon Textract - Amazon Web Services* [Amazon Web Services, Inc.]. (s.f.). Consultado el 7 de agosto de 2023, desde https://aws.amazon.com/es/textract/

Goodfellow, I. J., Bengio, Y., y Courville, A. C. (2016). *Deep Learning*. MIT Press. Consultado el 27 de julio de 2023, desde http://www.deeplearningbook.org/

Gorski, N., Anisimov, V., Augustin, E., Baret, O., y Maximov, S. (2001). Industrial bank check processing: The a2ia CheckReaderTM. *International Journal on Document Analysis and Recognition*, *3*(4), 196-206.

Graves, A. (2012). *Supervised Sequence Labelling with Recurrent Neural Networks* (Vol. 385). Springer.

Graves, A., Fernández, S., Gomez, F., y Schmidhuber, J. (2006). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. *Proceedings of the 23rd international conference on Machine learning*, 369-376.

Hashizume, A., Yeh, P.-S., y Rosenfeld, A. (1986). A method of detecting the orientation of aligned components. *Pattern Recognit. Lett.*, *4*(2), 125-132.

Krizhevsky, A., Sutskever, I., y Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. En P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou y K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States* (pp. 1106-1114).

Lu, Y., y Tan, C. L. (2003). Improved Nearest Neighbor Based Approach to Accurate Document Skew Estimation. *7th International Conference on Document Analysis and Recognition (ICDAR 2003), 2-Volume Set, 3-6 August 2003, Edinburgh, Scotland, UK*, 503-507.

*LUISA*. (s.f.). Consultado el 2 de julio de 2023, desde https://mh.udelar.edu.uy/luisa/

Martínek, J., Lenc, L., y Král, P. (2020). Building an efficient OCR system for historical documents with little training data. *Neural Computing and Applications*, *32*(23), 17209-17227.

Nagy, G., Seth, S. C., y Viswanathan, M. (1992). A Prototype Document Image Analysis System for Technical Journals. *Computer*, *25*(7), 10-22.

Nesmachnow, S., y Iturriaga, S. (2019). Cluster-UY: Collaborative scientific high performance computing in uruguay. En M. Torres y J. Klapp (Eds.), *Supercomputing* (pp. 188-202). Springer International Publishing.

Nguyen, T. T. H., Jatowt, A., Coustaty, M., y Doucet, A. (2021). Survey of Post-OCR Processing Approaches. *ACM Computing Surveys*, *54*(6), 124:1-124:37.

*Organismo Coordinador de Operaciones Antisubversivas (OCOA) | Sitios de Memoria Uruguay*. (s.f.). Consultado el 5 de julio de 2023, desde https://sitiosdememoria.uy/node/929

Papandreou, A., Gatos, B., Louloudis, G., y Stamatopoulos, N. (2013). ICDAR 2013 Document Image Skew Estimation Contest (DISEC 2013). *12th International Conference on Document Analysis and Recognition, ICDAR 2013, Washington, DC, USA, August 25-28, 2013*, 1444-1448.

Pavlidis, T., y Zhou, J. (1992). Page segmentation and classification. *CVGIP Graph. Model. Image Process.*, *54*(6), 484-496.

Postl, W. (1986). Detection of linear oblique structures and skew scan in digitized documents. *In Proceedings of the 8th International Conference on Pattern Recognition*, 687-689.

Rawls, S., Cao, H., Kumar, S., y Natarajan, P. (2017). Combining Convolutional Neural Networks and LSTMs for Segmentation-Free OCR [ISSN: 2379-2140]. *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, *01*, 155-160.

*SciPy*. (s.f.). Consultado el 8 de julio de 2023, desde https://scipy.org/

Smith, R. (2007). An Overview of the Tesseract OCR Engine [ISSN: 2379-2140]. *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, *2*, 629-633.

*TensorFlow*. (s.f.). Consultado el 3 de julio de 2023, desde https://www.tensorflow.org/?hl=es-419

*Tesseract user manual* [Tessdoc]. (s.f.). Consultado el 2 de julio de 2023, desde https://tesseract-ocr.github.io/tessdoc/

*tesstrain* [original-date: 2018-04-27T10:30:54Z]. (2023, 6 de julio). tesseract-ocr. Consultado el 18 de julio de 2023, desde https://github.com/tesseract-ocr/tesstrain

Wick, C., Reul, C., y Puppe, F. (2020). Calamari - A High-Performance Tensorflow-based Deep Learning Package for Optical Character Recognition. *Digital Humanities Quarterly*, *014*(2).

# ANEXOS

# Anexo 1

# Entrenamiento de OCR

## Entrenamiento de Tesseract

La instalación y ejecución de Tesseract para reconocer texto no presenta mayores complicaciones. Sin embargo, a la hora de entrenar un modelo la situación es más compleja. Para empezar, el proceso solo es soportado oficialmente en Linux según la documentación oficial.

Es necesario primero que nada, instalar ciertas dependencias que pueden encontrarse en la documentación oficial «Tesseract User Manual», s.f.

Los datos que se necesitan como entrada son pares de imágenes y un archivo de texto con la transcripción (imagen.tif e imagen.gt.txt). El formato de imagen fue TIFF pero otros formatos también son soportados. Sin embargo, es necesario realizar un preprocesamiento con anterioridad para generar a partir de estos archivos los archivos .box y .lstm que son los que utiliza Tesseract para entrenar.

Una vez qe se tienen estos archivos, es necesario dividir el conjunto de datos en archivos de entrenamiento y validación, guardando el listado de imágenes en cada conjunto en archivos de texto.

Finalmente, es necesario generar un diccionario con los posibles caracteres que se encuentran en las transcripciones.

Afortunadamente, existe una herramienta oficial llamada tesstrain que facilita todos estos pasos «tesstrain», 2023. Esta herramienta facilita mucho el entrenamiento de Tesseract ya que contiene scripts de Python para generar los archivos .lstm y .box. Además, contiene un archivo Makefile con una gran cantidad comandos de forma que el entrenamiento sea tan fácil como ejecutar make train.

Esta herramienta además permite configurar hiper parámetros como el paso de aprendizaje, cantidad máxima de iteraciones o seleccionar un modelo para hacer finetuning.

Durante la investigación realizada se desarrolló una imagen de Docker que contiene la herramienta tesstrain e incluye todas las dependencias necesarias para entrenar. En consecuencia, para entrenar Tesseract alcanza con tener las imágenes junto a su transcripción y ejecutar una imagen de Docker.

## Entrenamiento de Calamari

A diferencia de Tesseract, el proceso de entrenamiento con Calamari no presenta grandes dificultades. No obstante, la instalación puede resultar un tanto compleja.

En caso de utilizar un equipo con GPU, que es lo recomendado para reducir el tiempo de ejecución, es necesario instalar y configurar correctamente CUDA de Nvidia. Calamari utiliza TensorFlow y cada versión de TensorFlow soporta un rango de versiones de CUDA. En el caso de la versión de Calamari utilizada (2.2.2) se utilizó la versión 2.4.1 de TensorFlow con CUDA 11.5. Es importante revisar que todo esté correctamente instalado ya que de otra manera tanto la inferencia como entrenamiento van a ser muy lentos.

Para llevar a cabo el entrenamiento, se empleó la interfaz de línea de comandos (CLI) proporcionada por Calamari, específicamente el comando 'calamari-train'. Se hicieron uso de las siguientes opciones: '–trainer.output_dir' para especificar la

carpeta de salida del modelo, '–warmstart.model' en los casos en los que se realizó ajuste fino (finetuning), '–device.gpus' para seleccionar el uso de GPU en el entrenamiento, '–trainer.gen SplitTrain' para generar la partición de entrenamiento y '–trainer.gen.validation_split_ratio' para determinar la proporción del conjunto de validación respecto al conjunto de entrenamiento. Además, se utilizó la opción '–train.images' para indicar la ubicación de las imágenes y los archivos con las transcripciones.

Similar a Tesseract, se emplearon imágenes en formato TIFF. Para cada imagen, su correspondiente transcripción llevaba el mismo nombre, pero se modificaba la extensión de .tif a .gt.txt.

# Anexo 2

# Base de datos Mongo

En este anexo se plantea describir la base de datos generada durante el desarrollo de la tesis. Como fue mencionado, la misma es una base de datos documental MongoDB. Se optó por esta base de datos debido a que no hay relaciones entre las diferentes entidades, además de que no se conocía el esquema de los datos al momento de comenzar y MongoDB brinda flexibilidad. En la Figura A1 se encuentra un diagrama que contiene las colecciones y los campos que contiene cada documento. Se pueden dividir las colecciones en dos grupos: las que contienen los resultados de ejecutar los OCR y las que contienen las etiquetas con las que fueron entrenados los OCR. A continuación se describe el contenido y los campos de cada una de las colecciones.

La colección **fichas_ocr** contiene los resultados de los OCR sobre los campos de las fichas. *rollo* y *ficha* permiten identificar la imagen. *field* representa el campo (por ejemplo 'ci' o 'primer_nombre'), *pattern* indica la clase de la ficha (resultado de la clasificación) y finalmente los resultados de Tesseract y Calamari se encuentran en *ocr_tesseract* y *ocr_calamari* respectivamente.

Los resultados del OCR sobre el índice se encuentran en la colección **index_ocr**. En esta ocasión, los campos necesarios para identificar una imagen son: *rollo*, *imagen*, *linea* y *columna*. El valor de *linea* es un número entero que representa la línea dentro de la hoja del índice (según fue detectada por las herramientas desa-

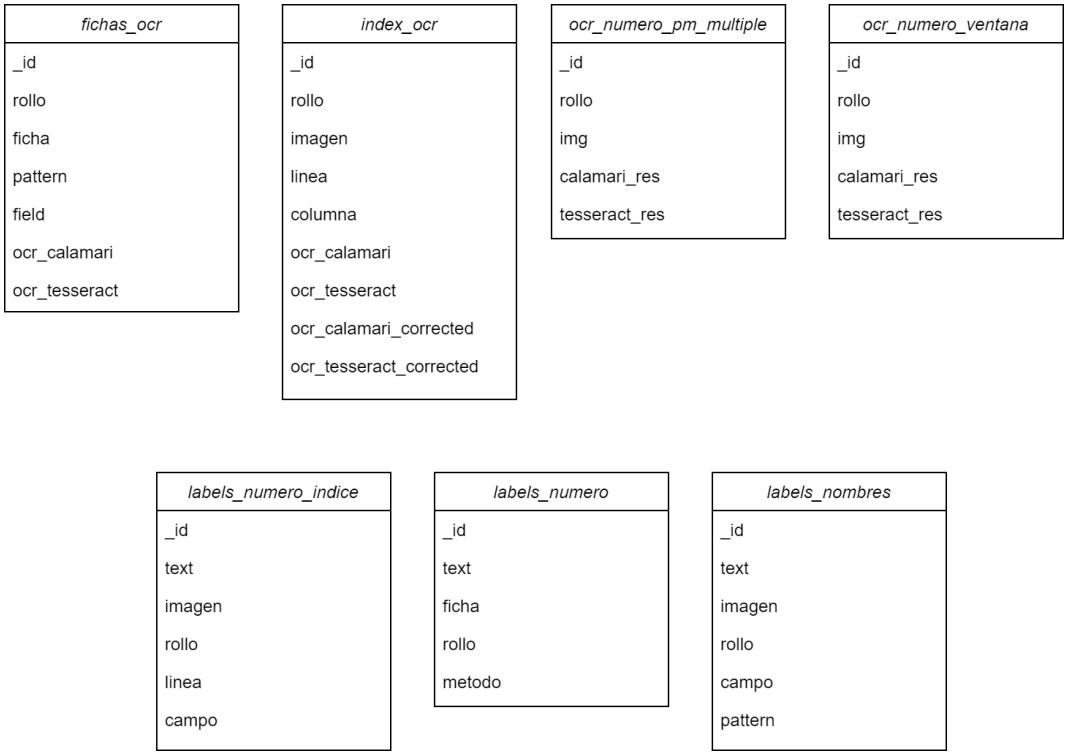



**Figura A1:** Base de datos MongoDB generada durante el desarrollo de la tesis.

rrolladas) mientras que el valor de *columna* es uno de los siguientes posibles: 'nro', 'nombre', 'alias', 'org_1', 'nro_org_1', 'org_2', 'nro_org_2', 'nro_carpeta', 'observaciones'. Finalmente, los resultados del OCR se encuentran en *ocr_tesseract* y *ocr_calamari*. Además, se suman dos nuevos campos: *ocr_tesseract_corrected* y *ocr_calamari_corrected* que contienen los resultados de los OCR corregidos.

Finalmente, las últimas dos colecciones con resultados de OCR son **ocr_numero_pm_multiple** y **ocr_numero_ventana** que contienen los resultados de aplicar OCR sobre el número de ficha, cada colección contiene los datos de uno de los dos métodos de extracción utilizados. *rollo* e *img* permiten identificar una imagen. *tesseract_res* y *calamari_res* contiene los resultados de aplicar Tesseract y Calamari.

Las etiquetas utilizadas para entrenar los OCR se almacenaron en las restantes tres colecciones. Las mismas tienen en común un campo *text* cuyo contenido es la etiqueta. Los datos restantes permiten identificar la imagen.

La colección **labels_numero_indice** contiene los resultados de etiquetar las co-

lumnas de nombre y número del índice. Los campos *imagen*, *rollo* y *linea* identifican la imagen y *campo* representa la columna. En este caso, 'numero' o 'nombre'.

Las etiquetas de los números de ficha extraídos de las fichas están en la colección **labels_numero**. *ficha* y *rollo* identifican la imagen. *metodo* contiene el método con el que se extrajo el número, hay dos posibles valores: 'ventana' y 'multiple'.

La última colección es **labels_nombres**. Esta colección contiene resultados de los datos etiquetados de las fichas. La ficha se puede identificar mediante *imagen* y *rollo*. *pattern* contiene la clase de la ficha (resultado de la clasificación) y finalmente, el campo etiquetado se identifica con *campo*. En este caso, los valores posibles son 'apellido_paterno.png', 'primer_nombre.png' y 'ci_.png' (ya que solo se etiquetaron esos campos).