UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

TÍTULO: Búsqueda de titulares históricos de dominio en folios reales digitalizados no indexados

AUTORES: APU Sebastian Joray

DIRECTOR/A: Prof. Javier Díaz

CODIRECTOR/A: Mg. Paola Amadeo

ASESOR/A PROFESIONAL: Lic. María Soledad Antonetti

CARRERA: Licenciatura en Sistemas Plan 2015

Resumen

En el marco de la modernización del Estado, se propone un análisis exhaustivo de las imágenes generadas por el proceso de digitalización de las inscripciones de dominios del Registro de la Propiedad de la Provincia de Buenos Aires, a fin de obtener una base de datos sobre la cual realizar búsquedas y proporcionar información necesaria relacionada a personas sobre las cuales se solicita saber un historial de titularidad sobre inmuebles.

Palabras Clave

LSTM - Tesseract - Optical character recognition - Computer Vision - TensorFlow - Folios Reales - ElasticSearch - Python - Keras

Conclusiones

Con las nuevas herramientas de visión por computadora y bases de datos NoSql, el problema de búsqueda de texto en archivos en papel se fue volviendo más fácil de solucionar, permitiendo resolver consultas que antes podían llevar meses, en sólo segundos. Otro punto a tener en cuenta es la aparición de las herramientas de IA para satisfacer este tipo de requerimientos, que no estaban presentes en el momento de la propuesta de la presente tesina.

Trabajos Realizados

- Investigación y relevamiento del estado del arte para este tipo de tareas.
- Desarrollo de una aplicación para la extracción de texto de las matrículas ya digitalizadas.
- Diseño de una base de datos NoSql.
- Comparación de diferentes métodos para entrenar modelos basados en LSTM con distintas tipografías.
- Desarrollo de un sitio de búsqueda a través de una aplicación Web accesible para los usuarios internos al organismo

Trabajos Futuros

- *Diseñar un subsistema para la administración de los pedidos.*
- *Pasar a producción del prototipo para un entorno de escala masiva.*
- *Comparar la eficacia de lo desarrollado con nuevas herramientas basadas en IA.*

Fecha de la presentación: MAYO 2024

Agradecimientos

Me gustaría expresar mi sincero agradecimiento a quienes han sido fundamentales en mi camino. A mi familia, agradezco su constante apoyo y comprensión, en especial a mi padre que ya no está presente, pero tenía una deuda con él.

Agradezco a mis amigos Flor, Cristian y Juan por incentivarme a seguir. A Sole y Marian, les agradezco por su incondicionalidad.

Mi reconocimiento especial va para Paola Amadeo, quien ha sido un faro de paciencia y ha compartido conmigo el entusiasmo por aprender. A Javier Diaz por las oportunidades brindadas.

Índice

Capítulo I. Introducción....3
Objetivos....8
Desarrollos propuestos....9
Organización del documento....9
Capítulo II. Elicitación de requerimientos....11
Historias de usuario....11
Capítulo III. El proceso de reconocimiento de caracteres OCR....15
Redes neuronales....15
LSTM en los algoritmos de detección de texto....16
Funcionamiento general del algoritmo de reconocimiento....19
Pre procesamiento de la imagen....20
Análisis de las diferentes librerías de OCR....22
Entrenamiento de la tipografía utilizando Tesseract....34
Capítulo IV. Identificación de la información según su ubicación....41
Análisis de la estructura de la matrícula....41
Herramientas de procesamiento del lenguaje natural (NLP)....45
Capítulo V. Base de datos documentales....50
Comparación de las bases de datos documentales....50
Operaciones sobre el índice ElasticSearch....53
Capítulo VI. Arquitectura propuesta....57
Aplicación web consumidora....59
Aplicación para el procesamiento de imágenes....60
Capítulo VII. Funcionamiento de la aplicación....62
Resultados y Análisis de las Pruebas de Usuarios del RPBA....67
Capítulo VIII. Conclusiones....68
Anexo 1....71
Anexo 2....76
Referencias Bibliográficas....83

Capítulo I. Introducción

En el marco de la modernización del Estado, nos situamos en la Dirección Provincial del Registro de la Propiedad Inmueble de la provincia de Buenos Aires [1]. Se propone un análisis exhaustivo de las imágenes generadas por el proceso de digitalización de las inscripciones de dominios de dicho organismo, a fin de obtener una base de datos sobre la cual realizar búsquedas y proporcionar información necesaria relacionada a personas sobre las cuales se solicita saber un historial de titularidad sobre inmuebles.

Como bien se dice en la tesina de grado de Javier Bautista [2]: “La incorporación del servicio de compra venta de inmuebles a través de una minuta electrónica [3], la solicitud de publicidad por ventanilla virtual, la comunicación electrónica de medidas cautelares y la creciente demanda a los servicios que presta cotidianamente el RPBA a la comunidad, determinó la necesidad de replantear el objetivo de los procesos haciendo uso de nuevas tecnologías”. A modo de complemento de dicho trabajo, nos enfocaremos, ya no en el formato de las entradas de información al organismo, sino en los procesos de búsquedas internos. Al momento de investigar las propiedades que ha poseído históricamente un sujeto, el organismo no posee una herramienta informatizada para hacerlo. Sólo se cuenta con una consulta llamada Índice de Titulares, que es actualizada por Agencia de Recaudación de la Provincia de Buenos Aires [4], pero que solamente contiene las titularidades actuales, es decir, no tiene en cuenta el historial.

En el Registro de la Propiedad existen 15 áreas de registración. Cada una tiene a cargo 1, 2 o varios partidos de la provincia de Buenos Aires. La mayoría de ellas son de Folio Real [5]. El resto son de folio personal o cronológico, que excede el alcance de este trabajo, ya que están escritas a mano y en un futuro serán convertidas por personal especializado a folio real. Un ejemplo de área de folio real es el Área 5, que posee los partidos de General Pueyrredón y Balcarce.

General Pueyrredón, con Mar del Plata como su mayor localidad, tiene alrededor de 275.000 inscripciones (números de matrícula), y cada una tiene al menos una hoja doble faz. A su vez, cada inscripción puede tener unidades funcionales lo que agrega una matrícula más, del tipo “B” (o planilla), que será una especie de matriz donde se anotarán las correspondientes unidades funcionales.

Estas unidades contienen otras matrículas en el caso de ser transferidas, que pueden tener varias carillas anotadas. Todo esto nos da un cálculo del orden de los millones de imágenes, para una sola área. Esto hace que una búsqueda manual sea casi imposible. Por lo cual se lleva adelante un proceso de digitalización, que permite el uso de nuevas herramientas para el análisis de dichas imágenes [6]. El proyecto, anterior a la presente tesina, denominado Registro Digital, tiene por objetivo contar con las matrículas de los inmuebles en formato digital en lugar de en papel. El proyecto se inició en el año 2015 y hasta el momento se han digitalizado tres partidos: Dolores, Castelli y Adolfo G. Chaves. Se realizaron escaneos masivos de las matrículas y se implementaron nuevos sistemas como el Índice de Matrículas, el Visor de Folio Real electrónico y el Administrador de Ficheros. Para entender de qué se trata el proyecto, hay que tener en cuenta algunos conceptos claves, como son las Unidades Funcionales (UF), Unidades complementarias (UC) y parcelas, que serían porciones o partes de un dominio completo, propiedad o inscripción. También se hace referencia a planillas B y PHB, que son tablas cuyas celdas son parcelas y UF/UC, respectivamente.

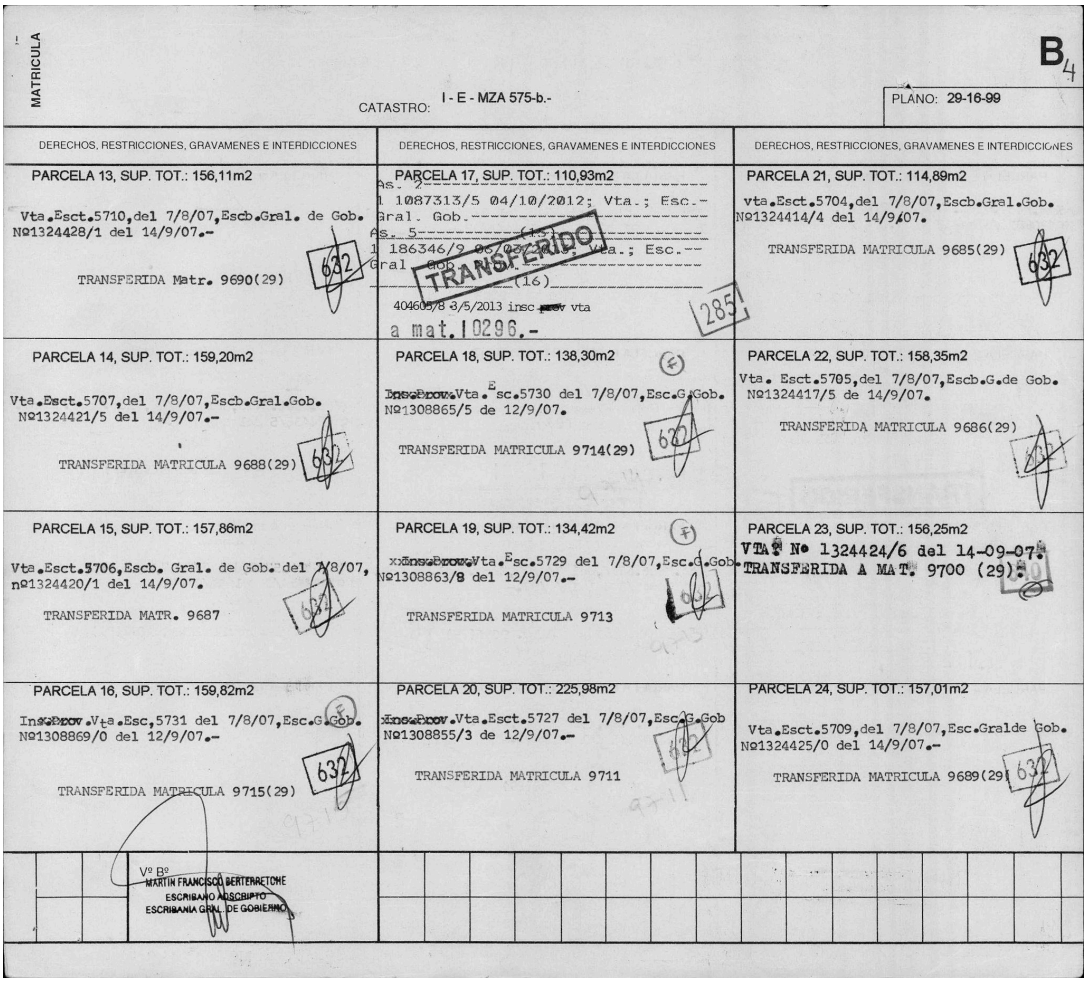
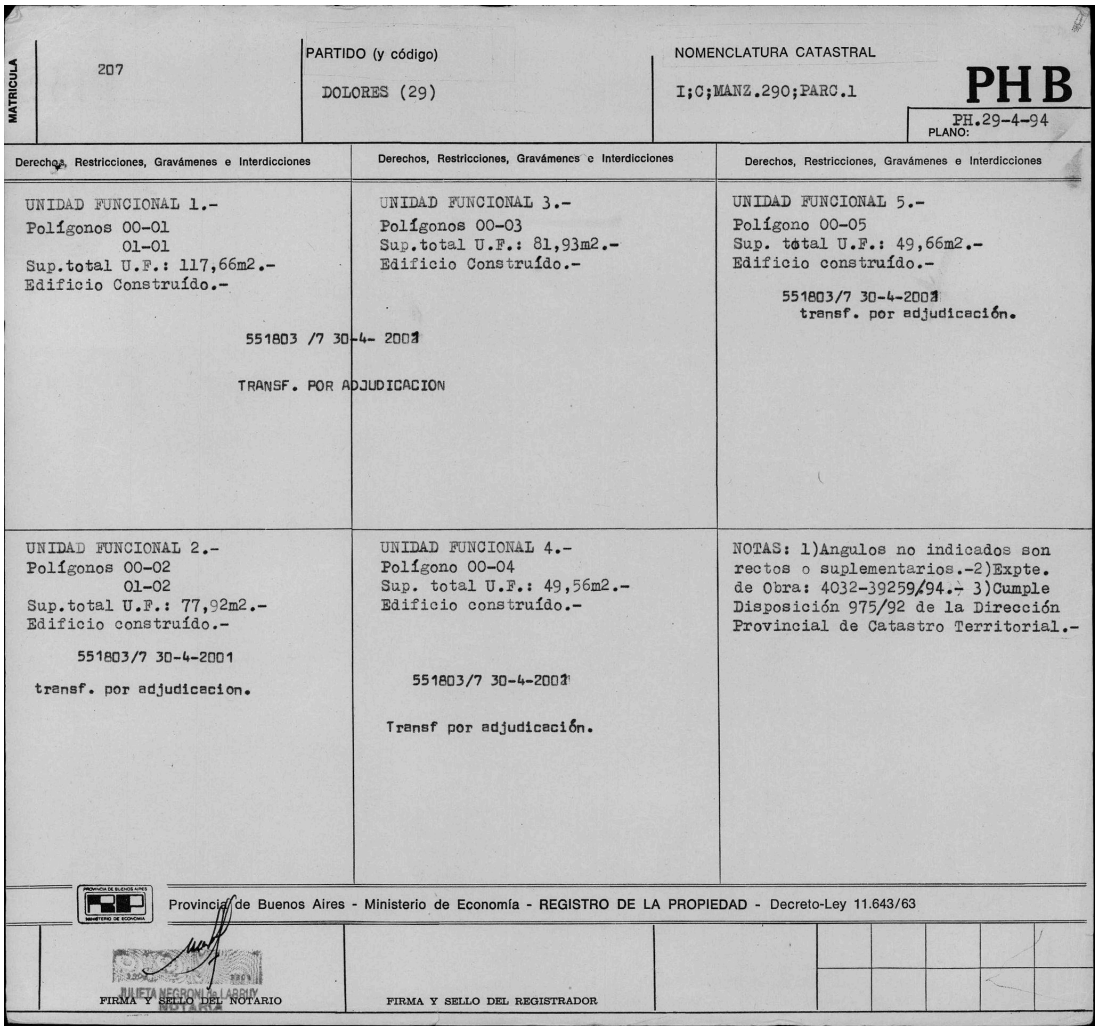


Fig. 1 - Ejemplo de planilla B

Como vemos en la figura 1, cada parcela del dominio original da lugar a una nueva matrícula, y por eso generan un desprendimiento del dominio original. En cambio, las planillas PHB determinan Unidades Funcionales o Complementarias, que siguen perteneciendo al dominio original, aunque tengan titulares diferentes (figura 2).



MATRICULA 207 | PARTIDO (y código) DOLORES (29) | NOMENCLATURA CATASTRAL I;C;MANZ.290;PARC.1 | PHB | PLANO: PH.29-4-94

Derechos, Restricciones, Gravámenes e Interdicciones

UNIDAD FUNCIONAL 1.-
Polígonos 00-01
01-01
Sup.total U.F.: 117,66m2.-
Edificio Construído.-

551803 /7 30-4- 2001

TRANSF. POR ADJUDICACION

Derechos, Restricciones, Gravámenes e Interdicciones

UNIDAD FUNCIONAL 3.-
Polígonos 00-03
Sup.total U.F.: 81,93m2.-
Edificio Construído.-

Derechos, Restricciones, Gravámenes e Interdicciones

UNIDAD FUNCIONAL 5.-
Polígono 00-05
Sup. total U.F.: 49,66m2.-
Edificio construído.-

551803/7 30-4-2001
transf. por adjudicación.

UNIDAD FUNCIONAL 2.-
Polígonos 00-02
01-02
Sup.total U.F.: 77,92m2.-
Edificio construído.-

551803/7 30-4-2001

transf. por adjudicacion.

UNIDAD FUNCIONAL 4.-
Polígono 00-04
Sup. total U.F.: 49,56m2.-
Edificio construído.-

551803/7 30-4-2001

Transf por adjudicación.

NOTAS: 1)Angulos no indicados son rectos o suplementarios.-2)Expte. de Obra: 4032-39259/94.- 3)Cumple Disposición 975/92 de la Dirección Provincial de Catastro Territorial.-

Provincia de Buenos Aires - Ministerio de Economía - REGISTRO DE LA PROPIEDAD - Decreto-Ley 11.643/63

FIRMA Y SELLO DEL NOTARIO

FIRMA Y SELLO DEL REGISTRADOR

Fig. 2 - Planilla PHB

Las imágenes obtenidas a través del proyecto de Registro Digital previamente descripto, forman el insumo básico del desarrollo llevado adelante en la presente tesina.

Además del Registro Digital mencionado previamente, encontramos otros antecedentes de destacar que se mencionan a continuación. Caseres, Delia, Thomas et al, en “Generation and Use of a Digest System by Integrating OCR and Smart Searches” [7] presentan una investigación realizada por la Universidad Nacional de La Plata sobre tecnologías similares a las que se aplican en este documento. Se analizan diferentes herramientas y sistemas utilizados en otras universidades y empresas para la gestión de la información, la comunicación y la colaboración. Se destaca la importancia de la implementación de estas tecnologías en la UNLP para mejorar la eficiencia y la calidad de los servicios ofrecidos a la

comunidad académica y a la sociedad en general. El trabajo propone la utilización de una base no-sql, llamada Apache Solr, basada en Lucene, para la búsqueda de documentos de un digesto de la Facultad de Ciencias Económicas de la UNLP. También propone Tesseract como herramienta de OCR.

Eduardo Xamena y Ana Gabriela Maguitman en “Language modeling tools for massive historical OCR post-processing” [8], hablan del tratamiento post-processing del resultado del proceso de extracción de texto, para analizar la salida del proceso de reconocimiento de caracteres y corregir errores. El objetivo principal es mejorar la calidad de los documentos digitalizados y reducir el tiempo y costo de la corrección manual de errores. El informe describe los diferentes métodos y herramientas, incluyendo la segmentación de texto, la corrección ortográfica y gramatical, y la detección y corrección de errores de formato. También se discuten los desafíos y limitaciones del proceso de post procesamiento, como la variabilidad en la calidad de los documentos digitalizados y la complejidad de algunos idiomas.

Rebecca Osborne y Catherine Scott en “Technical Guidelines for Digitizing Cultural Heritage Materials: Creation of Raster Image Master Files” [9], hablan sobre los tipos de imágenes y sus características. Si bien el documento es del 2010, proporciona directrices para la digitalización de imágenes fijas, incluyendo fotografías, dibujos y documentos. Se describen los diferentes tipos de imágenes y sus características, como la resolución, el formato de archivo y la profundidad de bits. También se discuten los requisitos de hardware y software para la digitalización y se proporcionan recomendaciones para la conservación y el almacenamiento de las imágenes digitales.

En su trabajo “Corrección automática de errores de OCR en documentos semiestructurados” [10], Pablo A. Paliza se basa en un trabajo de OCR del “Archivo Provincial de la Memoria de Córdoba” (APM), creado en 2006 por la legislatura de la Provincia de Córdoba a través de la Ley 9286, llamada “Ley de la Memoria”, adaptándolo para alfabetos no estándares o diferentes tipos de fuentes. También muestra que es posible entrenar estos motores con textos similares o una fracción del corpus que se quiere procesar, utilizando teoría de la probabilidad.

Las investigaciones previas presentadas por Caseres et al., Xamena y Maguitman, Osborne y Scott , y Paliza ofrecen perspectivas valiosas para el diseño

de un sistema integral de gestión de documentos. En primer lugar, se destaca la necesidad de un tratamiento especializado para las imágenes, considerando aspectos como la resolución y el formato de archivo. La elección de herramientas de OCR entrenables, como Tesseract, se recomienda para adaptarse a alfabetos no estándares y diversas fuentes, mejorando la versatilidad del proceso. El post-procesamiento del texto obtenido del OCR, centrado en la corrección de errores y formato, emerge como una etapa esencial para mejorar la calidad del material digitalizado. Por último, la implementación de un motor de búsqueda basado en Lucene se sugiere para una indexación eficiente y una recuperación de información optimizada. En conjunto, estas conclusiones proporcionan una guía práctica para el desarrollo de un sistema integral que garantice la eficiencia y precisión en la gestión de documentos.

De los estudios mencionados previamente sumados a la entrevista llevada a cabo con la coordinadora del proyecto de digitalización, asesora de la tesina, donde se analizan los proyectos planteados y las necesidades existentes, se propone el desarrollo de un software complementario al existente, que consiste en buscar dentro del texto de las imágenes, los nombres de personas que pudieran ser titulares de dominio, u otros derechos reales (hipotecas, usufructos, afectación a vivienda) en las imágenes escaneadas, que no se estaría resolviendo de forma automática.

A continuación, se presentan los objetivos del presente trabajo.

Objetivos

Objetivo general

- Habilitar medios de exploración de documentos legales no sistematizados para la búsqueda de información, a través de herramientas de visión computarizada y redes neuronales, utilizando motores de búsqueda basados en JSON.

Objetivos específicos

- Investigar herramientas para el reconocimiento de caracteres (OCR) en textos tipeados con máquinas de escribir antiguas, en idioma español.

- Investigar y comparar las diferentes formas de almacenar la información recolectada por el proceso de OCR (bases de datos de documentos).
- Construir una aplicación para brindar información requerida por algún organismo jurídico o por el mismo Registro de la Propiedad de la provincia de Buenos Aires, que busque coincidencias en nombres de personas dentro de las matrículas (folios reales). Esto nos permitirá construir una historia causal para las personas en cuestión.
- Buscar metadatos en las imágenes que nos pueden ayudar a descartar la información que no es relevante a nuestra búsqueda, como nombre de calles, partidos, etc., que puedan ser confundidas con personas buscadas.

Desarrollos propuestos

- Investigar y comparar los diferentes métodos para entrenar modelos con nuevas tipografías (en este caso serían máquinas de escribir antiguas).
- Desarrollar una aplicación para reconocimiento de caracteres de las matrículas ya digitalizadas para un perfil de administrador.
- El diseño de una base de datos que sea flexible (NoSQL) y tenga una rápida respuesta, como ElasticSearch o MongoDB.
- Desarrollar un sitio de búsqueda a través de una aplicación Web accesible para los usuarios internos al organismo.

Organización del documento

El capítulo I presenta el estado del arte y los objetivos de la tesina. El capítulo II, la etapa de elicitación de requerimientos del proyecto. En el capítulo III se hace una comparación de las diferentes alternativas que hay en el mercado para los módulos del proyecto y de cómo funciona el algoritmo de reconocimiento de caracteres, y se plantea algunas precondiciones para su utilización. También en este capítulo se profundiza sobre las redes neuronales que emplea el algoritmo para procesar las imágenes, que también servirían para trabajos futuros. En el capítulo IV se plantean estrategias para estructurar la información obtenida de las matrículas. En el capítulo V se comparan las diferentes opciones para almacenar la información generada por la etapa de reconocimiento de texto. En el capítulo VI se

elige un framework de MVC, para implementar el desarrollo propuesto. En el capítulo VII se habla del circuito administrativo y la forma en que se solicita la información. En el capítulo VIII se presentan las conclusiones y los trabajos a futuro que podrían realizarse.

Capítulo II. Elicitación de requerimientos

Para la elicitación de requerimientos se utilizó la técnica de historias de usuarios. Esta técnica es una práctica común en el desarrollo de software, particularmente en metodologías ágiles como Scrum [11]. Consiste en describir una funcionalidad o característica de un sistema desde la perspectiva del usuario final. Cada historia de usuario representa un pequeño fragmento de funcionalidad que aporta valor al usuario y se presenta en un formato simple y comprensible.

En la aplicación se identifican los siguientes tipos de usuarios:

1. El usuario externo:

   Es la persona que solicitará la información, a través del portal web del RPBA, y realizará un pago por ello. Podrán ser abogados, escribanos, o cualquier persona que acredite interés legítimo.

2. El usuario interno:

   Es la persona que procesará la consulta del usuario externo, y emitirá un resultado. Para ello deberá consultar las matrículas correspondientes y hacer una verificación ***ocular*** de la búsqueda realizada.

3. El administrador de Sistemas:

   Son usuarios pertenecientes al Departamento de Sistemas. Serán los encargados de ejecutar la aplicación de OCR. Para ello deben indicar la ubicación dentro del sistemas de archivos donde se localizan las imágenes que desea digitalizar, y elegir el modelo de OCR que desee utilizar.

Historias de usuario

i. **Nombre de la historia de usuario**: Solicitar búsqueda de matrículas relacionadas con Juan Pérez

Descripción: Como usuario externo a los servicios web del Registro de la Propiedad, deseo poder realizar una búsqueda de matrículas relacionadas con

Juan Pérez. Esto me permitirá acceder a la información de propiedades inmuebles vinculadas a esta persona y obtener detalles relevantes sobre las mismas.

Criterios de aceptación:

1. Como usuario externo, debo poder iniciar sesión en la página de servicios del registro de la propiedad utilizando mis credenciales válidas.
2. Una vez autenticado, debo tener acceso a la función de “búsqueda de titulares históricos en folio reales” en la página de servicios.
3. Al seleccionar la opción de búsqueda, se debe presentar un campo para ingresar el nombre "José Gómez" como término de búsqueda.
4. El sistema debe validar que el término de búsqueda ingresado sea válido y contenga al menos dos palabras.
5. Al enviar la solicitud de búsqueda, el sistema debe procesar la solicitud y mostrar un mensaje de confirmación de que la solicitud ha sido enviada exitosamente.

ii. **Nombre de la historia de usuario**: Procesar solicitud de búsqueda.

Descripción: Como usuario interno al Registro de la Propiedad de Buenos Aires (RPBA), deseo tener la capacidad de seleccionar una solicitud de búsqueda de matrículas pendiente de la lista y proceder a procesarla. Al procesar la solicitud, realizaré la búsqueda correspondiente en el sistema interno del RPBA y prepararé un informe con los resultados para enviar al usuario solicitante.

Criterios de aceptación:

1. Como usuario interno del RPBA, debo poder acceder a una lista de solicitudes de búsqueda de matrículas pendientes para su procesamiento.
2. La lista de solicitudes pendientes debe mostrar información relevante sobre cada solicitud, como el nombre del usuario solicitante y la fecha de la solicitud.
3. Debo tener la opción de seleccionar una solicitud específica de la lista para procesarla.

4. Al seleccionar una solicitud de búsqueda, debo recibir una confirmación para asegurarme de que deseo procesar esa solicitud en particular.

iii. **Nombre de la historia de usuario**: Realizar búsqueda de matrículas para un titular en cuestión.

Descripción: Como usuario interno del Registro de la Propiedad de Buenos Aires (RPBA), deseo tener la capacidad de realizar una búsqueda de matrículas relacionadas con una persona utilizando el sistema de búsquedas del RPBA. Además, necesito la funcionalidad para guardar los resultados de la búsqueda y enviárselos al usuario.

Criterios de aceptación:

1. El usuario interno confirma la selección y procede a utilizar el sistema de búsquedas del RPBA para buscar la información relacionada con la persona utilizando el nombre proporcionado en la solicitud.
2. El sistema de búsquedas muestra los resultados de la consulta con detalles de las matrículas relacionadas con el nombre proporcionado.
3. El usuario revisa los resultados para verificar la precisión de la información obtenida.
4. Si todo está correcto, el usuario genera un informe con los detalles de las matrículas encontradas.
5. El usuario envía el informe al usuario solicitante por e-mail.
6. La solicitud de búsqueda se marca como "procesada" y se elimina de la lista de pendientes.
7. El sistema registra la fecha y hora del procesamiento de la solicitud.

.

iv. **Nombre de la historia de usuario**: Actualizar los índices y generar archivos PDF (Portable Document Format en inglés), el cuál es un formato de documento portátil ideado por Adobe, que permite tener varias páginas.

Descripción: Como administrador de Sistemas del Registro de la Propiedad de Buenos Aires (RPBA), deseo tener la capacidad de procesar las imágenes en una determinada ubicación dentro de un recurso compartido, extrayendo el texto de

las imágenes, generando archivos de formato PDF con las mismas, y actualizando el índice de búsqueda.

Criterios de aceptación:

1. El administrador de Sistemas elige la ruta del recurso donde se encuentran las imágenes.
2. El administrador de Sistemas elige el modelo de reconocimiento a utilizar.
3. El sistema va informando de los avances del proceso, que puede llevar varios minutos.
4. El sistema emite un mensaje de finalización.
5. El administrador de Sistemas corrobora que los archivos PDF han sido creados.

Capítulo III. El proceso de reconocimiento de caracteres OCR

Las redes neuronales constituyen una herramienta fundamental para el proceso de OCR. A continuación, se presenta una breve descripción de las mismas, su uso para el proceso de OCR a través de la implementación de distintas librerías. Se analizan distintas elecciones de las disponibles en el mercado y la selección de una de ellas para su entrenamiento en el caso presentado de reconocimiento de texto en las matrículas del RPBA abordado en la presente tesis.

Redes neuronales

Como ocurre otras tantas veces, se está hablando de un desarrollo que nació hace décadas, pero que no ha tenido las aplicaciones deseadas hasta ahora. En el caso de las redes neuronales artificiales, hay que remontarse hasta 1943. Entonces, los investigadores Warren McCulloch y Walter Pitts dieron vida a la 'Lógica Umbral'.

Considerado como el primer modelo informático que usaba redes neuronales, marcó un antes y un después. Tras la Lógica Umbral, se abrieron dos vías de investigación. Por un lado, la que estudia los procesos biológicos del cerebro. Por otro lado, la creación de modelos de Inteligencia Artificial.

El siguiente hito histórico en el ámbito tecnológico corresponde a Alan Turing y su "máquina tipo B". Los estudios sobre las redes neuronales humanas y las artificiales se desarrollaban en paralelo y se retroalimentan. Era el principio de lo que hoy conocemos como Machine Learning. A continuación, se presentarán las redes neuronales de tipo recurrentes utilizadas en la presente tesina.

Redes neuronales recurrentes (RNR - many to many)

Las redes neuronales recurrentes se consideran buenas en el procesamiento sensible al contexto y para reconocer patrones que ocurren en series temporales. Sin embargo, las redes neuronales recurrentes tradicionales no han mostrado un rendimiento competitivo en tareas a gran escala como el OCR y el reconocimiento de voz, quizás debido al problema del gradiente de fuga. La

arquitectura Long Short Term Memory [12] fue diseñada para superar este problema. Alex Graves, Marcus Lewiki y otros [13] presentaron arquitecturas de LSTM bidireccionales para acceder al contexto en ambas direcciones: hacia adelante y hacia atrás. Imaginemos estar leyendo un texto y queremos comprender el significado de una palabra. La arquitectura bidireccional permite ver las palabras anteriores y las palabras siguientes para tener un contexto completo. Luego, ambas capas (direcciones) de la arquitectura LSTM se conectan a una sola capa de salida. Esto significa que la información procesada en las dos direcciones se combina en una sola salida final.

Para evitar la necesidad de tener datos de entrenamiento segmentados, Graves utilizó un algoritmo que trabaja hacia adelante y hacia atrás para alinear las transcripciones con la salida de la red neuronal. Esto le permitió utilizar datos de entrenamiento que no estaban previamente segmentados o divididos en partes específicas.

Para OCR impreso, las líneas de texto vienen en muchos tamaños diferentes, sin embargo, la posición relativa y la escala de los caracteres es una característica importante para distinguir los caracteres en escritura latina y una variedad de otras escrituras. Graves et al. introdujeron un proceso de normalización para escalar las líneas de texto escritas a mano con una velocidad de escritura variable y un sesgo no uniforme. Para OCR impreso, parece que se requieren diferentes tipos de normalización para un buen rendimiento.

Rashid et al. [14] dividió las líneas de texto en regiones ascendente, descendente y media, para el escaneo de líneas de texto.

LSTM en los algoritmos de detección de texto

Como mencionamos previamente, LSTM (Long Short-Term Memory) es una variante de las redes neuronales recurrentes (RNN) que se utiliza para modelar secuencias de datos, como texto, audio o series temporales. A diferencia de las RNN estándar, las LSTM tienen la capacidad de capturar y mantener información a largo plazo, permitiendo un mejor manejo de dependencias por más tiempo en los datos de entrada.

La idea central detrás de LSTM es utilizar unidades de memoria llamadas "celdas de memoria" para almacenar y actualizar información en un período de tiempo prolongado. Estas celdas de memoria están diseñadas para permitir la fluidez del gradiente a través del tiempo, lo que resuelve el problema de la desaparición o explosión del gradiente que puede ocurrir en las RNN estándar.

El proceso es intuitivo: para los datos que son secuenciales, como las acciones o el pronóstico del tiempo, los resultados anteriores pueden ser indicadores importantes para la próxima entrada y, por lo tanto, será beneficioso transmitir información de puntos de datos anteriores de manera consistente para nuevas predicciones. Del mismo modo, para la detección de letras en OCR, las letras detectadas previamente pueden ser útiles para determinar si la letra menos predicha en la línea tiene sentido (p. ej., si se detectan las letras "D" y "o", la "y" tiene una probabilidad mucho mayor de ser la siguiente letra en la línea que la "g", aunque pueden parecer similares).

La estructura básica de una celda LSTM consta de tres compuertas fundamentales:

1. Puerta de olvido (Forget Gate): Decide qué información anterior en la celda de memoria se debe olvidar y qué información se debe mantener.
2. Puerta de entrada (Input Gate): Decide qué nueva información se debe agregar a la celda de memoria.
3. Puerta de salida (Output Gate): Controla qué información de la celda de memoria se debe utilizar como salida de la celda LSTM.

   El esquema de una celda LSTM se muestra en la figura 3.

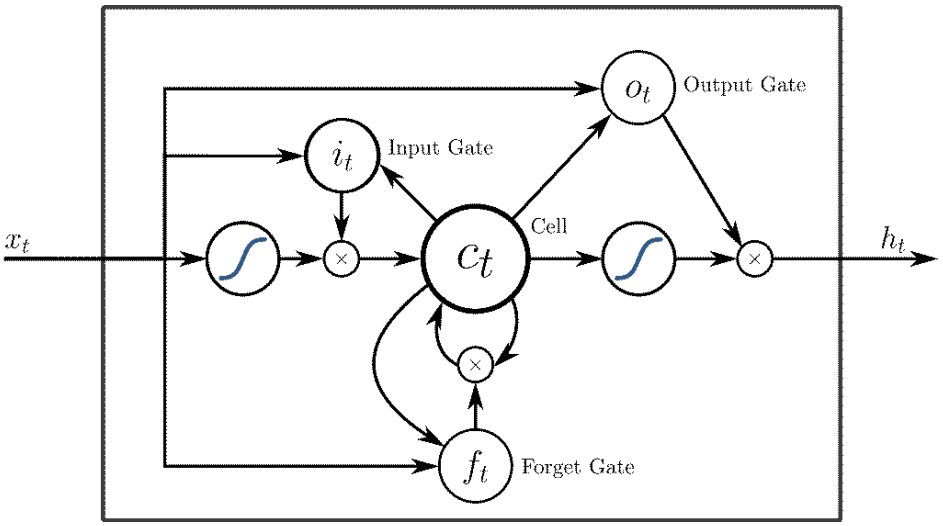


Fig.3 - Celda LSTM

Estas compuertas son redes neuronales sigmoideas que toman como entrada la información anterior y la información actual y generan valores entre 0 y 1. Estos valores determinan cómo se actualiza y fluye la información en la celda de memoria.

El proceso de cálculo en una LSTM implica la combinación de la información de entrada actual con la información anterior almacenada en la celda de memoria a través de las compuertas de olvido, entrada y salida. La información actualizada se pasa a la siguiente etapa de la secuencia y se repite este proceso para cada elemento en la secuencia.

Cada neurona o celda LSTM en la red se especializa en capturar y aprender diferentes patrones y características de los caracteres, incluyendo detalles como trazos, formas, curvas, entre otros. A medida que la secuencia de caracteres se propaga a través de la red LSTM, las neuronas se activan y generan salidas que contribuyen al proceso de reconocimiento de texto.

Es importante destacar que el aprendizaje de las neuronas LSTM se realiza a través del entrenamiento de la red con un conjunto de datos previamente etiquetados, lo que permite que la red identifique patrones y relaciones entre los caracteres para mejorar su capacidad de reconocimiento de texto.

Funcionamiento general del algoritmo de reconocimiento

A continuación, describiremos brevemente cómo funciona el algoritmo de reconocimiento de caracteres en herramientas basadas en LSTM. El orden en que se presentan es el orden secuencial en que se van realizando los procesos principales [15].

1. **Detección de los bloques**

El algoritmo en primer lugar dividirá la imagen de entrada en bloques de texto, áreas en blanco o gráficos.

2. **Búsqueda de líneas y palabras**

El siguiente paso es detectar unidades de texto: primero líneas y luego palabras. Se estima la altura de las líneas en cada región dada por la fase anterior, lo que permite filtrar el ruido y los caracteres que no son letras, cuya altura suele ser un pequeño porcentaje de la altura promedio de la línea.

3. **Reconocimiento de las palabras**

Posteriormente, las palabras se trocean en caracteres para proceder al reconocimiento aplicando técnicas especializadas para cortar caracteres unidos y asociar caracteres rotos.

4. **Clasificación de los caracteres**

En general, en primer lugar, se hace una lista corta con las clases de caracteres que el carácter desconocido podría ser en realidad. La lista se compone de las mejores coincidencias en una búsqueda de particularidades.

Luego, las particularidades del carácter se analizan y utilizan para calcular la similitud con los prototipos aprendidos.

5. **Análisis lingüístico**

Cuando el módulo de reconocimiento de palabras detecta una nueva segmentación, el módulo lingüístico (mal llamado “permutador”) elige la mejor cadena de palabras disponible en cada una de las siguientes categorías: palabra

más frecuente, palabra más frecuente del diccionario, palabra numérica principal, palabra MAYÚSCULAS principal, palabra superior en minúscula (con mayúscula inicial opcional) y palabra de elección del clasificador superior. La decisión final para una segmentación determinada es simplemente la palabra con la calificación de distancia total más baja, donde cada una de las categorías anteriores se multiplica por una constante diferente.

Pre procesamiento de la imagen

Para lograr un mejor resultado en el proceso de reconocimiento de caracteres (OCR) es necesario pulir o ajustar determinados atributos de la imagen, a modo de ser leídos más fácilmente por el software. Para ello utilizaremos la librería OpenCV(cv2) [16]. Esta librería open source es la más amplia en cuanto a funcionalidad ofrecida para visión por computadora y tiene licencia Apache 2.

A continuación, se presentan distintas funciones de esta librería que es recomendable aplicar a las imágenes, algunas de ellas utilizadas en el presente trabajo.

Transformar imagen a escala de grises

Esto se suele realizar cuando tenemos una imagen color en nuestro input. La función se utiliza de la siguiente manera, siendo el primer argumento la imagen original y el segundo el método a utilizar.

```
img_gris = cv2.cvtColor(image_color, cv2.COLOR_BGR2GRAY)
```

En el caso del presente proyecto, la imagen ya está en escala de grises, por lo cual este paso no se realiza.

Binarización de la imagen (thresh)

Luego de transformar la imagen, es necesario resaltar únicamente el texto de la misma y así eliminar algunas zonas donde no hay texto o fondo. Para hacer esto se debe realizar una binarización, que consiste en convertir la imagen de escala de grises a una imagen con solo dos tonalidades (0 y 1). En este caso específicamente llevamos a cabo una binarización por OTSU. Este tipo de binarización encuentra automáticamente un valor límite a partir del histograma de la

imagen y a todos los píxeles que se encuentren por debajo de ese valor le asigna 0; de lo contrario le asigna 1. En el siguiente ejemplo se presenta un ejemplo de invocación a la función donde el primer argumento es la imagen a trabajar, el segundo y el tercero representan los valores máximos y mínimos con los que trabajará la imagen y el último indica la binarización a realizarse en forma automática.

```
gray =
cv2.threshold(image,0,255,cv2.THRESH_BINARY|cv2.THRESH_OTSU)
```

Reducción del ruido

Se utiliza la función cv2.medianBlur() de OpenCV para aplicar un filtro de mediana a una imagen con el fin de reducir el ruido presente en la misma.

El ruido en una imagen se refiere a las variaciones no deseadas o artefactos que pueden aparecer debido a diversas razones, como la calidad de captura de la imagen, interferencia electromagnética o cualquier otro factor externo. El ruido puede afectar negativamente la calidad de la imagen y dificultar su procesamiento y análisis.

El filtro de mediana es un tipo de filtro espacial. Este filtro reemplaza cada píxel de la imagen con el valor de mediana calculado a partir de una vecindad definida alrededor del píxel. Al reemplazar el valor del píxel con su mediana local, el filtro de mediana es capaz de eliminar los valores atípicos o ruidos presentes en esa vecindad. En el siguiente ejemplo se presenta un ejemplo de invocación a la función, siendo image la imagen a trabajar y el segundo parámetro el tamaño del filtro de la ventana.

```
def remove_noise(image):
    return cv2.medianBlur(image,5)
```

Erosión

La erosión es un proceso morfológico utilizado en el procesamiento de imágenes para reducir el tamaño de los objetos o regiones presentes. Esta operación se basa en el uso de un elemento estructurante, que define la forma y el tamaño del objeto que se utilizará para "erosionar" o reducir los píxeles de la

imagen original. En el siguiente ejemplo se presenta un ejemplo de invocación a la función.

```
def erode(image):
    kernel = np.ones((5,5),np.uint8)
    return cv2.erode(image, kernel, iterations = 1)
```

En la función erode(image), se define un elemento estructurante llamado kernel utilizando np.ones((5,5), np.uint8). El kernel es una matriz 2D compuesta por unos (píxeles blancos) y tiene un tamaño de 5x5 píxeles. El tamaño del kernel determina el grado de reducción de los objetos en la imagen. A continuación, se aplica la operación de erosión utilizando

```
cv2.erode(image, kernel, iterations=1)
```

El primer argumento, image, es la imagen de entrada en la que se realizará la erosión. El segundo argumento, kernel, es el elemento estructurante utilizado para la erosión. El tercer argumento, iterations, indica el número de veces que se aplicará la operación de erosión. En este caso, se realiza una única iteración. La función devuelve la imagen resultante después de aplicar la operación de erosión. En esta imagen, los objetos o regiones presentes se reducirán y se separarán, dependiendo del tamaño y forma del kernel utilizado.

Erosión seguida de dilación

La apertura es una operación morfológica que combina dos operaciones básicas: erosión seguida de dilatación. Es útil para eliminar el ruido y pequeñas imperfecciones en los objetos de una imagen, al tiempo que mantiene la forma y estructura general de los objetos más grandes.

Análisis de las diferentes librerías de OCR

En esta sección se analizan las diferentes opciones de herramientas para el reconocimiento de caracteres más populares, realizando un estudio comparativo entre ellas según la documentación oficial y luego una prueba de concepto con dos imágenes testigo. Ambas imágenes pertenecen al partido de Bahía Blanca y fueron puestas a disposición por la asesora profesional. Se presentarán los resultados

obtenidos de aplicar el proceso de reconocimiento utilizando las diferentes herramientas y modelos y una comparación entre las mismas.

El desarrollo de la tesina se llevó a cabo utilizando Python como lenguaje de programación. Este lenguaje incluye distintas librerías para el reconocimiento de caracteres. Las opciones más populares se presentan a continuación:

- Keras OCR: Implementada con TensorFlow, que es la librería más popular para implementar redes neuronales en Python [17] [18]
- Paddle OCR: Es parte del proyecto PaddlePaddle , una librería china que fue avanzando rápidamente, la cual, si bien tiene buenos resultados aún en castellano, no logra superar en tiempo de ejecución a Tesseract. [19] [20]
- Tesseract OCR: Es la más antigua, y, por ende, la más entrenada. Fue implementada por Hewlett-Packard en 1984 y hoy es mantenida por Google. [21]

La tabla 1 presenta un análisis comparativo entre las herramientas de reconocimiento de caracteres mencionadas previamente.

Características	**Tesseract OCR**	**Paddle OCR**	**Keras OCR**
Lenguaje de programación	C++, Python, Java	Python	Python
Soporte de idiomas	Soporta una amplia gama de idiomas (es el más entrenado para español, por su antigüedad)	Soporta múltiples idiomas, incluyendo chino, inglés, francés, alemán, etc.	Depende de la configuración y el modelo utilizado, pero se puede adaptar a varios idiomas.
Precisión	Buena precisión en textos impresos y bien formateados, pero puede tener dificultades con fuentes o estilos de texto inusuales.	Buena precisión en textos impresos y manuscritos en varios idiomas, incluyendo caracteres chinos.	La precisión puede variar según los modelos y la calidad de los datos de entrenamiento.
Facilidad de uso	Puede requerir configuración y ajustes para obtener mejores resultados. No	Ofrece una API fácil de usar y una interfaz de usuario gráfica (GUI) para simplificar el	Puede requerir cierta experiencia en Python y Deep Learning para su implementación y

	tiene una interfaz de usuario gráfica (GUI) nativa.	proceso de reconocimiento de texto.	configuración.
Arquitectura	Basado en una clase particular de redes neuronales recurrentes llamadas LSTM (long short term memory) y el algoritmo de reconocimiento óptico de caracteres.	Utiliza arquitecturas de redes neuronales convolucionales (CNN) como ResNet y otros componentes para el reconocimiento de texto.	Utiliza modelos de redes neuronales convolucionales (CNN) y Recurrent Neural Networks (RNN) para el reconocimiento de texto.
Pre - entrenamiento	Utiliza modelos pre-entrenados en una amplia variedad de datos y se puede fine-tunear con datos adicionales.	Proporciona modelos pre-entrenados para textos impresos y manuscritos en varios idiomas, y también permite el fine-tuning con datos propios.	No proporciona modelos pre-entrenados específicos, pero se pueden utilizar modelos pre-entrenados de otras bibliotecas de Deep Learning.
Comunidad y Soporte	Tesseract OCR tiene una comunidad activa y una buena cantidad de documentación disponible.	PaddleOCR tiene una comunidad activa y es mantenido por PaddlePaddle, que es una plataforma de IA popular en China.	Keras OCR, al ser una biblioteca de código abierto basada en Keras, tiene una comunidad activa y hay una cantidad considerable de recursos y documentación disponibles.
Desarrollador / Mantenedor	Google	PaddlePaddle	Varios contribuyentes (Proyecto de código abierto)

Tabla 1 - Comparación de librerías de OCR, en base a la documentación oficial

Prueba de concepto con la primera imagen

La figura 4 muestra la imagen 1-1-1-1838-.tif, brindada por la asesora profesional. El nombre del archivo se corresponde a la organización interna del proyecto Registro Digital, sobre la cual se realizará la primera prueba de concepto, en la cual se han ocultado los apellidos de los titulares por cuestiones de privacidad.

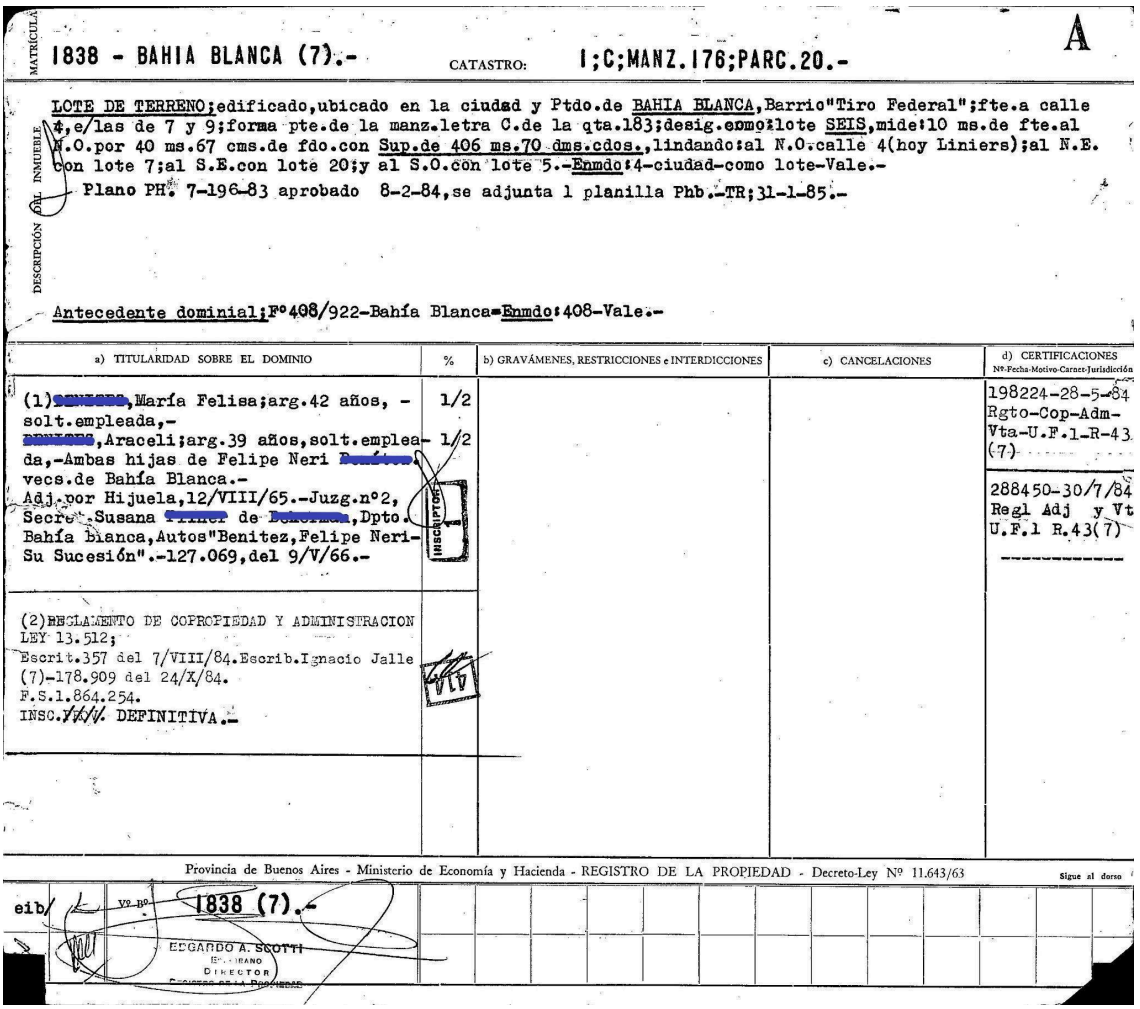


Fig. 4 - Matrícula 1838 de Bahía Blanca

Keras OCR

Para realizar la prueba de esta librería, se utilizó un notebook en Google Colab basado en el proyecto Keras, de Fausto Morales, ingeniero consultor egresado del MIT (Instituto Tecnológico de Massachusetts). Keras es una librería en Python que funciona bajo TensorFlow, librería de código abierto en C++, la cual se emplea para el desarrollo de redes neuronales y machine learning (ML).

Se aplicó el proceso de Keras a la figura 4. En texto resaltado en amarillo se indica los tres nombres de personas a ser identificadas:

- Be**tez, María Feliza;arg. 42 años
- Be**tez, Araceli;arg. 39 años
- Susana Pl**er de Per***man

Los dos primeros corresponden a titulares y uno a un funcionario interviniente (con algunos caracteres ocultos).

s a 77m 1838 bahia blanca icamanz izbparce2os catastros ptdosde bahia blancay barrio utiro federaln terreno sedificadoubicado ciudsd jitesa calle lote de 1a en y gtalszidesig cade eomoblote seiss mideslo gjforma pteade manzoletra la de ftesal 1as de 14 f5 t ss ef y e lindandoial noorcalle athoy linierssal supede 406 msazo amsacdoses nees oo 40 67 cmsade fdoscon por ms a seecon seenmdos lotie 2oiy sooacon lote amciudaducomo lotenvaleem lote tal al on phi 7219683 mtri plano aprobado bw2849se 31ml85m adjunta planila phb l e josi22bahia joswvaletm nialjfo blancauenmdo antecedente domi di certificaciones titularidad sobre el dominio gravamenes o by restricciones interdicciones cancelaciones a nofccha jurisdiccion motivo carnict 19622423is 112 ijben**ezs gmaria felisa 42 anoss saxge rgt ouncopma amn empleadas solt vtamuafluras 112 ben**ez garaceli jarge39 anoss solte emplea 674 benitez neri dagmambas hi jas de felipe bahia blanca vecside 84 3017 2884 50u hiuelaanitiosuzene adjn por regl adj vt y secren susana pl**er de bek***anidpto l ufel ra3177 biancagautos bahia n h benitez gfelipe neria z 9ivj66m ont su suc l2700699 del esi ejnoglatento coproptedad de y dmlnt cion al sira ley 131 5123 escri tvinbasesoribinacio t0357 del jalle 17178909 24jxjs4e del fe s18642 548 insceysnia defs initiva provincia de buenos aires ministerio de economia hacienda registro de la propiedad decretoley no y 11643 j63 siguc dorsa al 1838 c72s vlabo eib klu ergado af scott er peano deke c to e i ggs

La figura 5 muestra el marcado que hace Keras sobre el caso testigo, delimitando cada palabra o cifra que logró extraer.

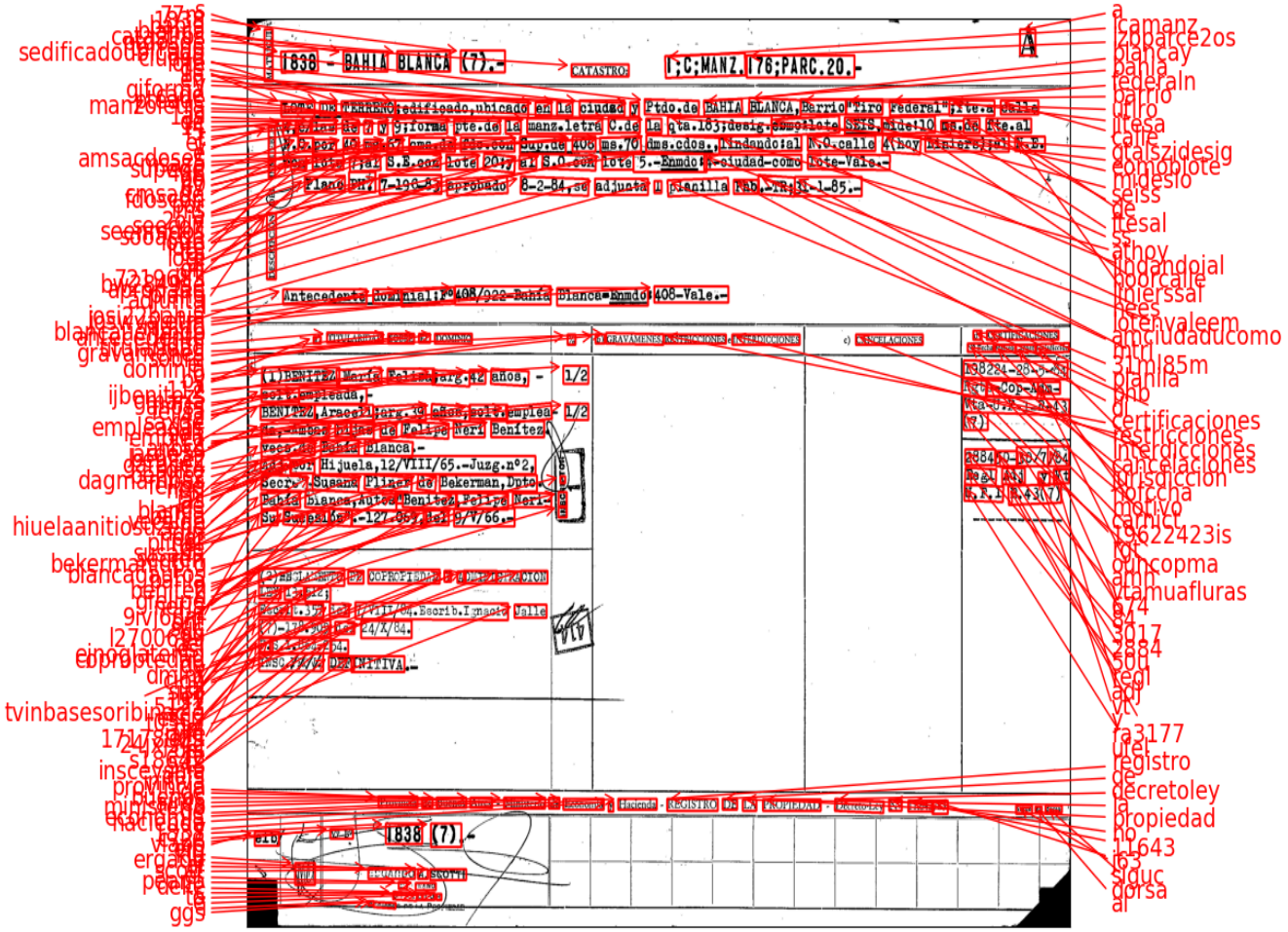


Fig. 5 - Matrícula 1838 intervenida con Keras

Lo que se puede observar también es que Keras no devuelve las palabras en forma ordenada y no reconoce la letra ñ, por ejemplo. Esto no quitaría la posibilidad de cierto post procesamiento, que se describe en el capítulo 5.

Paddle OCR

PaddleOCR es un kit de herramientas de OCR multilingüe basado en PaddlePaddle, y es un sistema de OCR de poca ocupación de recursos de procesamiento, admite reconocimiento de más de 80 idiomas, proporciona herramientas de síntesis y anotación de datos, admite entrenamiento e implementación dentro de servidores, dispositivos móviles, integrados y de IoT. PaddlePaddle (PArallel Distributed Deep LEarning) es un framework de aprendizaje

profundo de código abierto lanzado por Baidu en 2016. Ofrece una plataforma unificada para diversas tareas de aprendizaje automático.

A continuación, se presentará el resultado del proceso de extracción de texto de la figura 4 con PaddleOCR:

```
1838 - BAH1A BLANCA (77-
1:C:MANZ.176:PARC.20.-
CATASTRO:
LOTE DE TERRENO;edificado,ubicado en la ciudad y Ptdo.de BAHIA BLANCA,Barrio"Tiro Federal";fte.a calle
4,e/las de 7 y 9;forna pte.de la manz letra C.de la qta.183;desig.epmo.lote SEIS,nide*10 ms.de fte.al
W.0.por 40 ms.67 cms.de fdo.con Sup.de 406 ms.70-dms.cdos.,lindando:al N.0.calle 4(hoy Liniers);al N.E.
Plano PH* 7-196.83.aprobado 8-2-84,se a&junta 1 planilla Phb.-TR;31-1-85:-
DESCRIPCION
Antecedente dominial;Fo408/922-Bahia Blanca Enmdo:408-Vale:-
a) TITULARIDAD SOBRE EL DOMINIO
b) GRAVAMENES, RESTRICCIONES c INTERDICCIONES
c) CANCELACIONES
d) CERTIFICACIONES
%
N%-Fecha-Motivo-Carnet-Jurisdiccióa
1/2
198224-28-5-84
(1)BEN***Z,Maria Felisa;arg.42 aios,
Rgto-Cop-Adm-
solt.empleada,
Vta-U.F.1-R-43
BEN***Z,Araceli;arg.39 aos,solt.emplear
-7)-
da,-Ambas hijas.de Felipe Neri Benitez!
vecs.de Bahia Blanca
2884 50-30/7/84
Adj por Hijuela,12/VIII/65.-Juzg.n°2,
```

```
Regl Adj y Vt
Secret.Susana. Plin*r deBek***an,Dpto.(
U.F.1 R.43(7)
Bahia Bianca,Autos"Benitez,Felipe Neri-
Su Sucesi6n".-127.069,del 9/v/66.-
NOIOVHISINTHAY A CVCEIEOHIOO GI OLETVTEEE(Z)
LEY 13:512;
Escrit.357 del 7/VIII/84.Escrib.Ignacio Jalle
(7)-178.909 del 24/x/84.
F.S.1.864.254.
INSC.YKYY DEFINITIVA.
Provincia de Buenos Aires - Ministerio de Economia y
Hacienda - REGISTRO DE LA PROPIEDAD - Decreto-Ley N9
11.643/63
Sigue al dorso
1838
eib
EegArDo A. Sgott
.- IrANo
DIReCtor
```

Si bien los resultados obtenidos son alentadores, no reconoce caracteres de lenguas latinas como la ñ o los acentos.

Tesseract

Tesseract es un motor OCR desarrollado en Hewlett-Packard (HP) entre 1984 y 1994. Comenzó como una tesis doctoral en la Universidad de Bristol, publicada en 1987. El desarrollo de este proyecto se detuvo por completo en 1994. Al año siguiente, se presentó en la Prueba Anual de Precisión de OCR, donde obtuvo resultados competitivos frente a las soluciones comerciales de OCR existentes por entonces. El proyecto estuvo congelado hasta 2005, cuando HP decidió lanzarlo como código abierto. Google se hizo cargo principalmente del proyecto, que ahora está disponible en un repositorio de GitHub [22]. Tiene su origen en la implementación de LSTM basada en Python de OCRopus, pero se ha

rediseñado por completo para Tesseract en C++. Es anterior a TensorFlow, pero compatible.

El flujo del proceso de OCR que realiza Tesseract se puede observar en la figura 6, descripto previamente en forma general. Leptónica es una dependencia de Tesseract a través de la cual se brinda soporte para varios tipos de imágenes. También obtiene información sobre el posicionamiento y el diseño de la página.

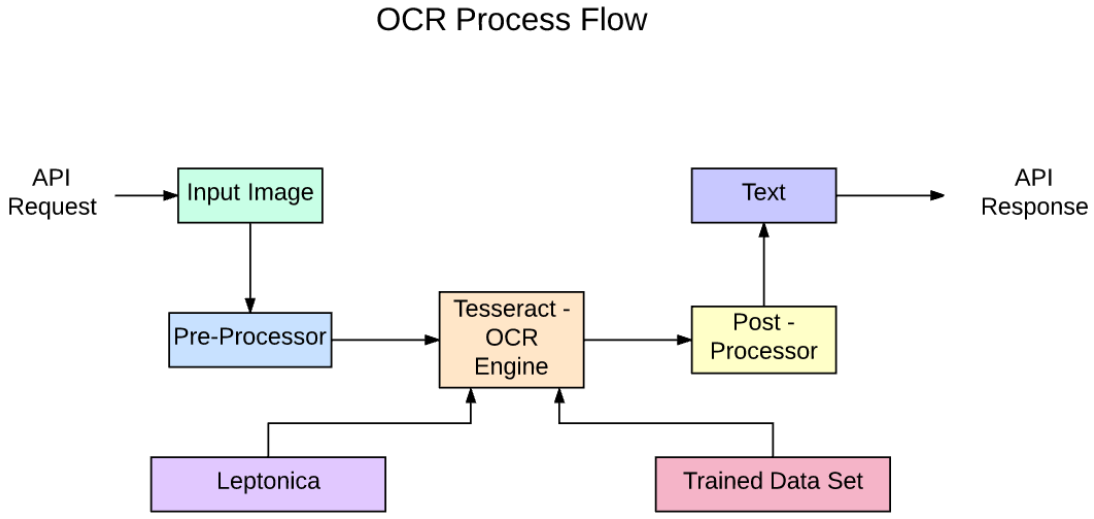


Fig. 6 - Flujo de Proceso de OCR

Una de las características más importantes de este OCR es su capacidad de aprendizaje. Después de un proceso de aprendizaje, Tesseract puede mejorar en la fase de reconocimiento para una fuente definida. En este caso Spa_old, que fue entrenado con máquinas de escribir antiguas.

Al ejecutar la prueba de concepto en la primera imagen utilizando el modelo Spa_old. el resultado fue el siguiente:

```
E 1838 - BAHIA BLANCA (7).- CATASTRO: |
;C;MANZ.178;PARC.20.-
la LOTE DE TERRENO ;¡edificado,ubicado en la ciudad y
Ptdo.de BAHIA BLANCA, Barrio"Tiro Federal";fte.a calle
= 4, e/las de 7 y 9;¡foraa pte.de la manz.letra Cede la
qta.l83;desig.anmoilote SElS,midellO ms.de ftecal
E \N.O.por 40 ms.67 cms.lde fdo.con Sup.de 406 ms.70
dms.cdos.,lindandotal N.O.calle 4(hoy Liniers)jal NE,
5 n lote 7Tzal S.B.con lote 20;y al S.O.con lote
5.-Enmdoi4-ciudad=como lote>ValeJ>-
```

| Plano PH, 7=-l96833 aprobado B-2-84,se adjunta l planilla Phb ¿<TR} 31-1-85.- Á

É

- Antecedente dominial¡Fo408/922-Bahía BlancasEnmdos 408-Vale +

; a) TITULARIDAD SOBRE EL DOMINIO % |» cara $, RESTRICCIONES e !N 1: amas | a canos |

'- 1 Net Fe:ds Moto Chrare |

"© (1) BEN***Z, María Felisajarg:42 años, = 1/2 198220295

solt.empleada,- Hi epáias

BEN***Z ,Aracelijarg.l39 años, solteemplea> 1/2 (7) efi

da,-Ambas hijas de Felipe Neri Benítez 7)

vecs8.de Bahía Blanca. Z

_Agj<vor Hi juela,l2/VI111/65.-Juzg.n°2, 2884 50-30/7/84

"Secru®.:Susana Pl**er de Bek***an, Dpto Regl Adj y vt

Bahía bianca, Autos"Benitez, Felipe Neri=[|% U.FPel R43(7)

Su Suc esién".-1l27.069, del 9/V/66.- Z <<< <=

(2) KB3LANETG DE CCFRCFISDAD E ASES RA CIC |

LEY 13. 5123

Ssexico357 dal TAVI1I 84 eEncrinel mnicic Jalle |

(7)=ii8e 509 del 24/K/Es.]

WoSele Cb 254.

1882. /Á// DEFINITIVA (>

Pravincia de Buenos Aires - Ministcrio de Economía y Hacicnda - REGISTRO DE LA PROPIKDAD - Decrctoles NES

SIISE

Podemos observar que identificó las ñ y caracteres particulares, aunque el carácter "*;*" lo confunde con la letra j.

Prueba con la segunda imagen

La figura 7 muestra otra imagen sobre la cual se realizarán las pruebas.

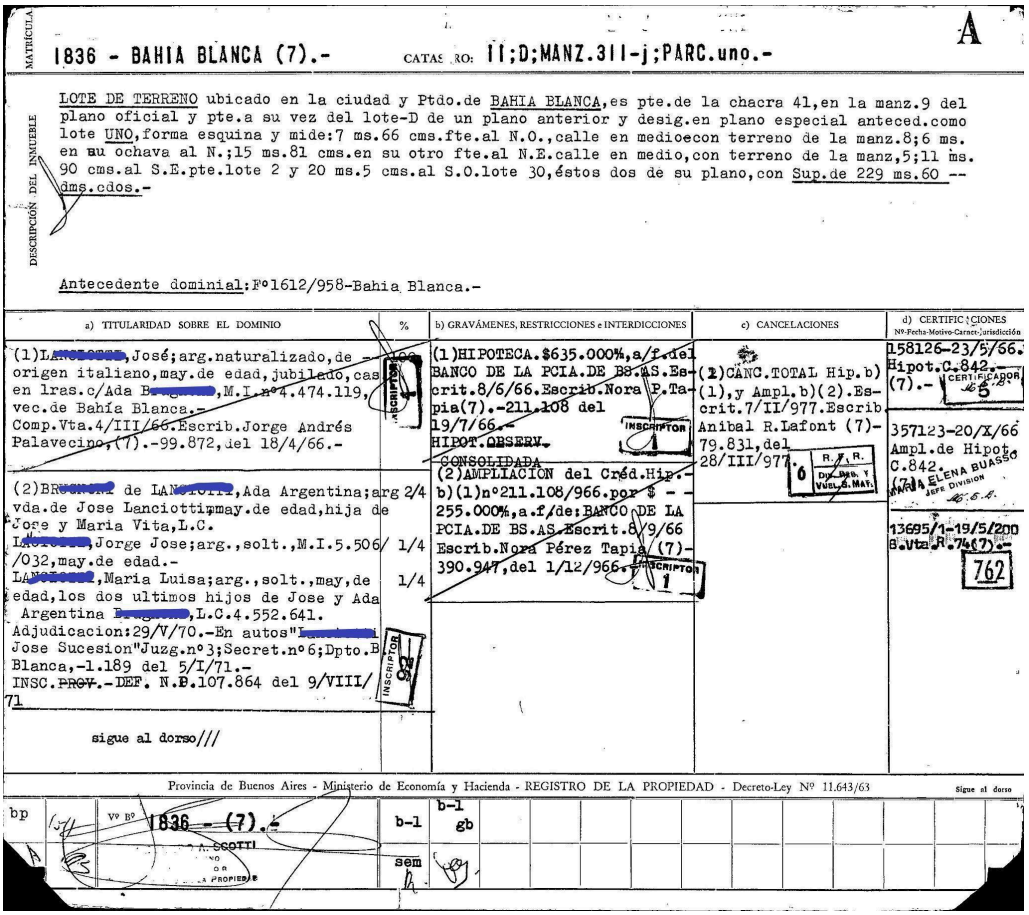


Fig. 7 - Matrícula 1836 de Bahía Blanca

En el Anexo II se muestran los resultados de aplicar los diferentes modelos a esta última imagen.

En la tabla 3 se muestra una tabla comparativa con determinados términos extraídos de la segunda imagen analizada. Para este caso se entrenó Tesseract con distintos modelos.

Texto original	**Keras**	**PaddleOcr**	**Tesseract Spa**	**Tesseract Eng**	**Tesseract Spa_old**
Lan***tti, José	Lan***tti Jose	Lan***rti José	Lan***tti Josézarg	Lan***tti José	La***tel, Joséjarg
Ada Bru***ni	Ada Bru***ni	Ada Bru***ni	Ada Bru***ni	Ada Bru***ni	Ada Bre***ni

María Luisa	Maria Luisai	Maria Luisa	Maria Luisajarg	María Luisa	Maria Luisajarg

Tabla 3 - Tabla comparativa de términos

Según este test, el que mayor tasa de aciertos tuvo, fue el modelo ENG (inglés de Tesseract). Esto se explica en que, si bien estamos en un contexto de lenguaje español, los apellidos de las personas no responden al lenguaje español, ya que no contiene caracteres especiales, y a su vez el modelo ENG es el más entrenado para este tipo de máquinas de escribir. Podemos suponer que es por antigüedad o frecuencia de uso.

Análisis comparativo según los resultados obtenidos de las pruebas de concepto

La librería Keras fue entrenada principalmente en lenguaje inglés. En el caso de la aplicación que se está desarrollando, sería interesante distinguir ciertos caracteres que son propios del lenguaje español, como son las vocales con acento prosódico, la letra ñ, y también los signos de interrogación de apertura “¿”, que, en el caso de los modelos entrenados exclusivamente para inglés, podrían ser confundidos con otro símbolo, y consecuentemente dificultar la búsqueda de determinados nombres o apellidos (José, García, etc)

En este sentido Tesseract está mejor entrenado. Al ser el de uso más extendido en el tiempo, contiene modelos exclusivos para el idioma español. Su uso se ha diversificado.

Por otro lado, tanto con Keras como con PaddleOCR, permiten localizar las palabras dentro de la imagen. Esto con Tesseract no sería posible.

Otra desventaja de Tesseract es la poca velocidad de procesamiento, justamente por ser modelos demasiado grandes, tarda más en aplicarlos.

Finalmente, con Tesseract no es posible obtener un remarcado de la palabra en la imagen, pero una persona avezada en el uso de folios reales, rápidamente encontraría al sujeto que se está buscando.

Si bien se vería la necesidad de optar por alguna librería particular, las alternativas mencionadas no necesariamente son excluyentes entre sí. Se podría realizar un enfoque secuencial utilizando cada una de las bibliotecas en forma

iterativa: realizando un reconocimiento con una de las librerías, seguido por otro utilizando alguna de las otras dos, y así sucesivamente. Dado que la base de datos está diseñada para el procesamiento a gran escala, existe la posibilidad de replicar los resultados para la imagen de una matrícula específica, permitiendo así que la misma esté repetida en nuestra base de datos. Es relevante destacar que los nombres no recuperados por una librería podrían ser recuperados por alguna de las otras dos, lo cual incrementa la probabilidad de encontrar algún término específico.

Además, es esencial considerar si los motores de OCR mantienen el orden de las palabras. Como se mencionó previamente, Keras tiene la particularidad de perder el orden en el que leemos, dificultando la aplicación de un post procesamiento para obtener metadatos.

En base a los estudios previos y mi experiencia laboral, el análisis comparativo y los resultados obtenidos en el curso de este trabajo, se ha optado por adoptar Tesseract como la opción para nuestras necesidades.

Entrenamiento de la tipografía utilizando Tesseract

A partir de la elección anterior, en esta sección de describirá la etapa de entrenamiento de la tipografía utilizando Tesseract. Como se ha dicho, la ventaja de Tesseract, es que al estar implementado con redes neuronales recurrentes LSTM, tiene la posibilidad de entrenar su modelo de reconocimiento para mejorar la tasa de aciertos para una determinada tipografía, dado un idioma establecido.

En este apartado se analizan distintos métodos para entrenar la tipografía y se plantean los principales desafíos de esta etapa. Existen dos razones fundamentales para entrenar la tipografía:

- Reconocer caracteres de una tipografía poco usual
- Utilizar un nuevo lenguaje

La aplicación objetivo de esta tesina se encuentra en el primer caso. O, mejor dicho, una combinación de ambos, puesto que son tipografías poco usuales (textos escritos en máquinas de escribir de diferentes modelos y años), y es un lenguaje poco explorado, porque como se puede imaginar, las herramientas fueron desarrolladas y entrenadas bajo el lenguaje inglés. Si bien el español o castellano

no es un lenguaje poco usual, la utilización de esta tecnología en países de habla hispana parece ser menos frecuente.

Entrenamiento a través de contenedores (Docker)

Una forma de entrenamiento es la presentada en el tutorial citado en Medium [23]. Este método utiliza contenedores de Docker [24], que es una plataforma diseñada para ayudar a los desarrolladores a construir, compartir y correr aplicaciones. Esto implica cierta complejidad, ya que requiere referenciar los contenedores, además de tener instalado el software necesario para manipular los mismos. También requiere tener imágenes recortadas junto con los textos correspondientes a esas imágenes, de modo que el modelo identifique los caracteres que en ellas se encuentren.

Por otro lado, al ser un método bastante artesanal, posibilita incluso entrenarlo con manuscrita, porque justamente, no utiliza una tipografía preexistente para que aprenda.

Entrenamiento con un notebook de Colab

A continuación, se presenta cómo entrenar una nueva tipografía de Tesseract en Google Colab [25]. Este método tiene las siguientes características:

- Se realiza a través de un notebook.
- No requiere instalación local.
- Se puede utilizar cualquier fuente de TrueType.
- Se realiza en pocos pasos.

La tipografía Old TypeWriter es una de las que más se ajusta a las máquinas que se utilizaron para confeccionar las matrículas, pero podría haberse optado por cualquier otra de esta familia.

En la figura 8 se observa la forma en que se utiliza este notebook.

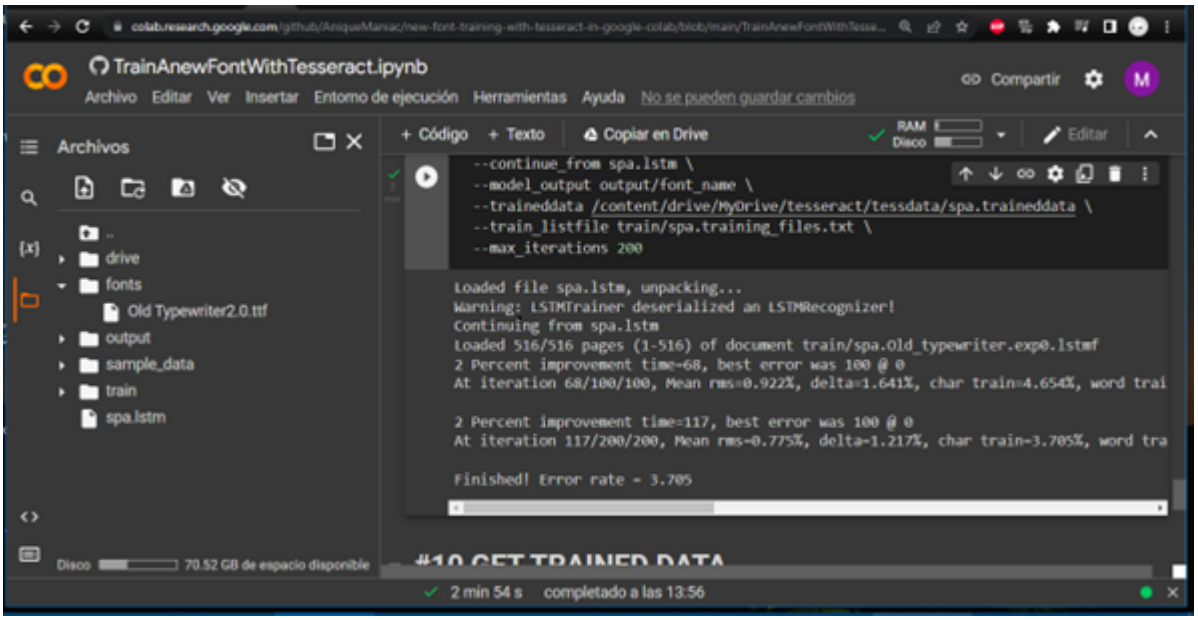


Fig. 8 - Google Colab para entrenar una nueva tipografía con Tesseract

Entrenamiento a través de un editor de box

El producto más conocido de este género es jTessBoxEditor [26] que se presenta en la figura 9. El mismo tiene las siguientes características:

- Proporciona edición de data box (áreas donde se encuentra el texto) de formatos Tesseract 2.0x y 3.0x y automatización completa del entrenamiento de Tesseract
- Puede leer imágenes de formatos comunes, incluido TIFF de varias páginas.
- Se requiere Java Runtime Environment 7 o posterior

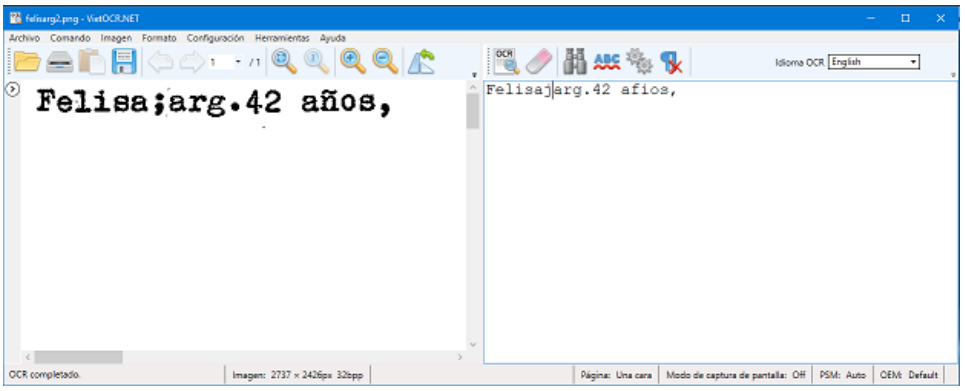


Fig. 9- jTessBoxEditor para entrenar una nueva tipografía con Tesseract

Podemos elegir los modelos disponibles según nuestra conveniencia, a saber:

- spa (español)
- eng (inglés)
- typewriter (modelo personalizado)

Luego de realizar distintas pruebas utilizando los distintos métodos, la notebook de Colab resultó ser la más conveniente por la posibilidad de usar cualquier fuente TrueType y por simplicidad de uso.

Elección del modelo de entrenamiento de Tesseract

Una vez seleccionado el método, procedemos a seleccionar el modelo de entrenamiento. Existen varios modelos de entrenamiento de Tesseract OCR disponibles para español, pero no todos están diseñados específicamente para reconocer texto escrito en máquinas de escribir antiguas. Sin embargo, existen algunos modelos que podrían ser útiles en este caso como spa_old, spa y ENG que se detallan a continuación.

Dado que cierto modelo identifica algunos titulares (personas) y tal vez otro identifica otros, podemos hacer diferentes pasadas del proceso de OCR a fin de mejorar los resultados, cambiando el modelo en cada pasada. Esto podría hacerse en el mismo script, o correr el script cambiando el modelo a utilizarse. Todo ello irá incrementando el repositorio de datos y se obtendrá más información a fin de aumentar la precisión de los resultados de la búsqueda, aunque haya redundancia.

Modelo de entrenamiento “Spa_old”

Este modelo ha sido entrenado con una amplia variedad de fuentes y tipografías utilizadas en las máquinas de escribir antiguas y puede producir buenos resultados en esta clase de texto. Es posible descargarlo desde repositorio `tesseract-ocr/langdata` de Github [27]. Este modelo ha sido entrenado para el idioma español con el motor antiguo de Tesseract, es decir, previo a la utilización de redes neuronales. El mismo contiene información sobre los caracteres del alfabeto latino, incluyendo las letras con tilde (á, é, í, ó, ú), la letra ñ y los signos de puntuación y acentuación propios del idioma (¿, ¡, «,»). Tiene una precisión menor que el modelo “spa” basado en LSTM, pero también requiere menos recursos computacionales para funcionar.

Para utilizar este modelo, se debe descargar desde el repositorio y compilar Tesseract OCR con el soporte para el idioma español. Luego, se debe indicar a Tesseract OCR que utilice este modelo para el reconocimiento del texto. Se puede realizar utilizando la opción `--oem 1` y especificar la ruta al archivo del modelo descargado con la opción `--tessdata-dir`.

Por ejemplo, para utilizar el modelo `Spa_old` en una imagen llamada `texto.tif`, podría ejecutarse el siguiente comando en la línea de comandos:

```
tesseract texto.tif output --oem 1 --tessdata-dir /ruta/al/directorio/del/modelo
```

Este comando utilizará el modelo `Spa_old` para realizar OCR en la imagen `texto.tif` y almacenará el resultado en un archivo de texto llamado `output.txt`. En cada caso se debe de reemplazar `/ruta/al/directorio/del/modelo` con la ruta real en el sistema donde se haya descargado y descomprimido el modelo `Spa_old`.

El modelo Spa_old **no puede ser entrenado** pues no está basado en LSTM.

Modelo de entrenamiento Spa

El modelo Spa está configurado y entrenado específicamente para procesar texto en español. Esto implica que ha sido optimizado para reconocer y comprender las características lingüísticas y gramaticales de este idioma, lo que mejora la precisión del OCR para trabajar con documentos escritos en esta lengua. El modelo Spa se puede obtener descargándolo del repositorio oficial de Tesseract. Utiliza la técnica LSTM, aunque tiene la posibilidad de ejecutarse en modo “legacy”. Esto significa que puede operar en un modo más antiguo o heredado si es necesario, posiblemente para mantener la compatibilidad con versiones anteriores de Tesseract u otros requisitos específicos.

En el caso de la aplicación desarrollada para esta tesina, este modelo mejoró al Spa_old, con la contrariedad de que tampoco reconoce correctamente el signo de puntuación punto y coma (;), aunque esto podría subsanarse aplicando otros filtros.

Modelo de entrenamiento personalizado

El proceso de entrenamiento personalizado generalmente consiste en alimentar al sistema con muestras de texto y luego ajustar el modelo de Tesseract para que sea más capaz de reconocer caracteres y patrones en esas tipografías particulares. Esto permite que la herramienta realice un reconocimiento de texto más preciso y efectivo en situaciones donde las mismas son usuales, como en la detección de matrículas de vehículos.

Para el caso de esta tesina, este modelo de entrenamiento resultó ser el más efectivo. El modelo original de Tesseract se ha adaptado para comprender y reconocer las tipografías que son similares a las utilizadas en las matrículas del RPBA. Para lograr esto, se han recopilado fuentes de tipografía TrueType (como "Old Typewriter" o "Typewriter.ttf") que se asemejan a las utilizadas en las matrículas.

En resumen, entrenar Tesseract de manera personalizada implica adaptar su capacidad de reconocimiento de texto para trabajar mejor con tipografías específicas, en este caso, tipografías similares a las utilizadas en folios reales o matrículas, con el objetivo de mejorar la precisión del reconocimiento.

Para entrenar el modelo Tesseract en español, se utilizó la tipografía llamada Typewriter, lo cual produjo la particularidad de reconocer que el símbolo ";" no es una letra "j", aunque a veces lo confunde con un signo de interrogación "¿". Sin embargo, esta confusión es preferible ya que el "¿" puede ser utilizado como separador en algunos contextos. Es decir, a nuestros fines, es útil. Lo vemos en el siguiente ejemplo de texto reconocido por la herramienta:

```
1 (1)B***EZ,María Felisa¿arg.42 años, - | 1/2 198224-28-5-64
```

En este caso, si bien el punto y coma ha sido reconocido con otro signo de puntuación, es útil debido a que separa sintácticamente la palabra Felisa de la palabra “arg.”, algo que no pasaría si lo reconociera como una letra j.

Finalmente es importante destacar que los procesos mencionados previamente hacen a la gestión completa del reconocimiento de caracteres de la imagen, siendo el tamaño de las mismas otro punto relevante a considerar.

Las imágenes proporcionadas por la asesora profesional son de formato .tiff en escala de grises y de un tamaño aproximado entre 500 kilobytes y 1 megabyte cada una. Esto fue una dificultad para la librería Image, que es la utilizada como primera opción en todos los proyectos de Python, provocando que no reconociera la compresión. Para solucionar este problema se utilizó la librería Scikit-Image [28]. Algo similar sucedió con la librería utilizada para crear los archivos PDF. Para solucionarlo se utilizó una parte de dicha librería **Io**, a fin de transformar las imágenes en archivos con formato JPG y luego confeccionar los archivos **PDF**.

Capítulo IV. Identificación de la información según su ubicación

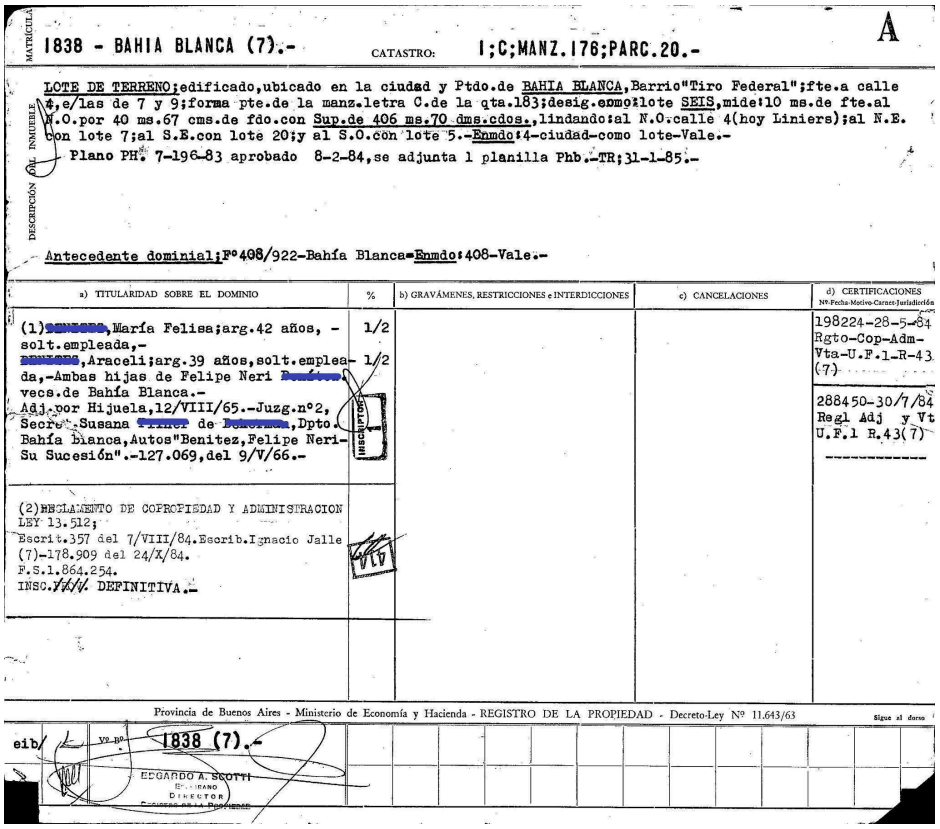
Una de las problemáticas que presentan las matrículas del RPBA radica en que los nombres de los propietarios pueden coincidir con nombres de calles, de localidades, entre otros. Por ejemplo, al apellido San Martín, el procesamiento de caracteres OCR puede identificarlo en la matrícula en diferentes lugares. Es relevante entonces poder discriminar entre aquellos que corresponden a nombres de titulares de aquellos que no lo son.

En este capítulo se describen distintas estrategias o formas de abordar el desglose de la información contenida dentro de las matrículas dada su relevancia para identificar el nombre de los titulares en forma unívoca.

Análisis de la estructura de la matrícula

Esta estrategia consiste en detectar líneas de referencia verticales y horizontales de la imagen que constituyen la matrícula. De esta forma se determinan los rectángulos donde debería estar el texto y finalmente se recorta la imagen para aplicar el OCR.

La figura 10 muestra la matrícula 1838 del Partido de Bahía Blanca sobre la cual se aplicará este proceso.


1838 - BAHIA BLANCA (7).- CATASTRO: I;C;MANZ.176;PARC.20.- A

MATRICULA

DESCRIPCIÓN DEL INMUEBLE

LOTE DE TERRENO;edificado,ubicado en la ciudad y Ptdo.de BAHIA BLANCA,Barrio"Tiro Federal";fte.a calle 4,e/las de 7 y 9;forma pte.de la manz.letra C.de la qta.183;desig.ofmo;lote SEIS,mide:10 ms.de fte.al N.O.por 40 ms.67 cms.de fdo.con Sup.de 406 ms.70 dms.cdos.,lindando:al N.O.calle 4(hoy Liniers);al N.E. con lote 7;al S.E.con lote 20;y al S.O.con lote 5.-Enmdo:4-ciudad-como lote-Vale.-
Plano PH. 7-196-83 aprobado 8-2-84,se adjunta 1 planilla Phb.-TR;31-1-85.-

Antecedente dominial;Fº 408/922-Bahía Blanca-Enmdo:408-Vale.-

a) TITULARIDAD SOBRE EL DOMINIO	%	b) GRAVÁMENES, RESTRICCIONES e INTERDICCIONES	c) CANCELACIONES	d) CERTIFICACIONES Nº-Fecha-Motivo-Carácter-Jurisdicción
(1) [redacted], María Felisa;arg.42 años, - solt.empleada,- [redacted], Araceli;arg.39 años,solt.empleada,-Ambas hijas de Felipe Neri [redacted] vecs.de Bahía Blanca.- Adj.por Hijuela,12/VIII/65.-Juzg.nº2, Secr.-Susana [redacted] de [redacted],Dpto.- Bahía blanca,Autos"Benitez,Felipe Neri- Su Sucesión".-127.069,del 9/V/66.-	1/2 1/2			198224-28-5-84 Rgto-Cop-Adm-Vta-U.F.1-R-43 (7)
(2)RECLAMENTO DE COPROPIEDAD Y ADMINISTRACION LEY 13.512; Escrit.357 del 7/VIII/84.Escrib.Ignacio Jalle (7)-178.909 del 24/X/84. F.S.1.864.254. INSC.XXXX DEFINITIVA.-				288450-30/7/84 Regl Adj y Vt U.F.1 R.43(7)

Provincia de Buenos Aires - Ministerio de Economía y Hacienda - REGISTRO DE LA PROPIEDAD - Decreto-Ley Nº 11.643/63 Sigue al dorso

eib/ Vº Bº 1838 (7).- EDGARDO A. SCOTTI DIRECTOR

Fig. 10- Matrícula ejemplo

La función erosión de la librería cv2 en Python "elimina" el texto de la imagen dejando solo las líneas verticales y horizontales (esto sólo funciona si la imagen de la planilla está alineada, sin una rotación ni deformación significativa). A partir de esta nueva imagen se podrían aplicar algoritmos para encontrar los recuadros, es decir, una lista de rectángulos (4 coordenadas en la imagen x1, y1, x2, y2). Utilizando esta información se podría identificar el tipo de planilla, las áreas donde existe texto y a cuál de los campos de la planilla se corresponde.

En un primer intento por localizar los titulares, se identificará la primera columna de la matrícula. La figura 11 muestra la imagen luego de aplicar el proceso de erosión.

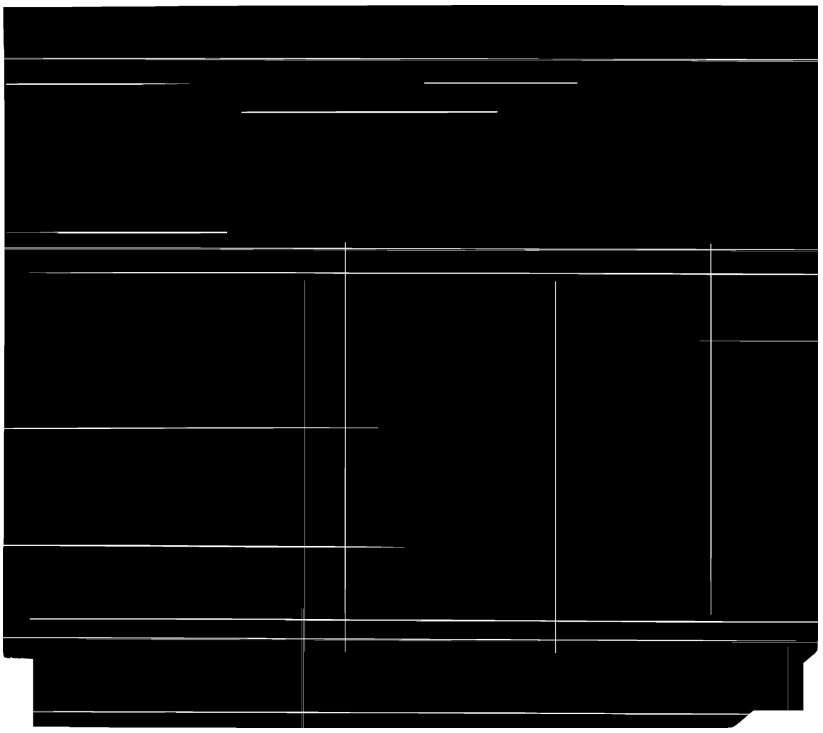


Fig. 11 - Imagen erosionada

El script nos arroja los siguientes datos:

```
LH: (0,180)->(2734)
LH: (0,815)->(2734)
LH: (0,898)->(2734)
LH: (0,2122)->(2734)
LH: (0,2366)->(2734)
LV: (1012,814)->(1347)
LV: (1149,793)->(2120)
LV: (1854,815)->(2370)
LV: (2376,798)->(2126)
```

Dicha lista muestra las coordenadas del origen de cada línea, y hasta dónde llega. En caso de ser horizontales, hasta qué valor de x en la pantalla. Y si son verticales, en qué valor de y (fila) de puntos en la pantalla.

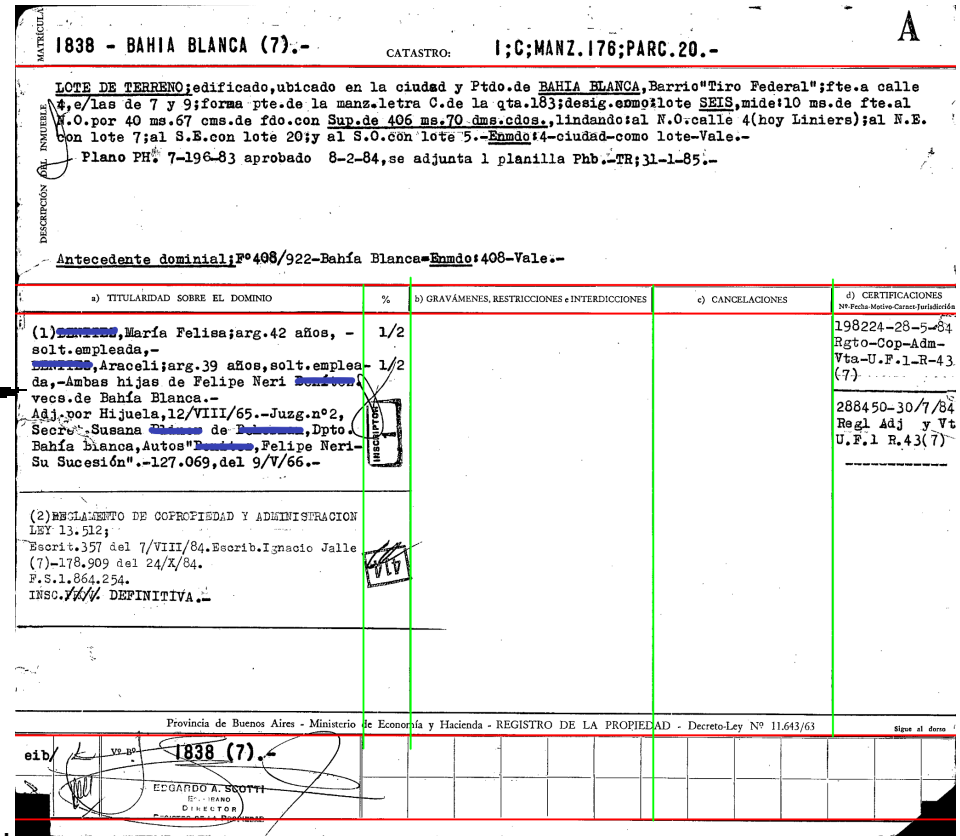


Fig. 12 - Imagen 1838 original superpuesta con la imagen erosionada

En la figura 12 podemos observar en color verde (líneas verticales) y rojo (líneas horizontales) las líneas que el script logró reconocer. Dicho script, que detecta las líneas horizontales y verticales y las dibuja sobre la planilla original, se encuentra en el anexo 1.

Una de las problemáticas detectadas con esta estrategia es que no podemos determinar luego en qué orden se encuentra el rectángulo que se debe localizar, ya que las matrículas tienen varios modelos diferentes, y también contienen líneas agregadas, hechas por los usuarios a modo de separación, que suman más rectángulos aún, con lo cual no se podría determinar, en principio, si el rectángulo con la información relevante es el octavo o no. Por ejemplo, en la figura 12, el rectángulo de titulares sería el octavo (se cuenta de izquierda a derecha, y de arriba hacia abajo). Pero, en la imagen 1836, sería el sexto, como se puede observar en la figura 13.

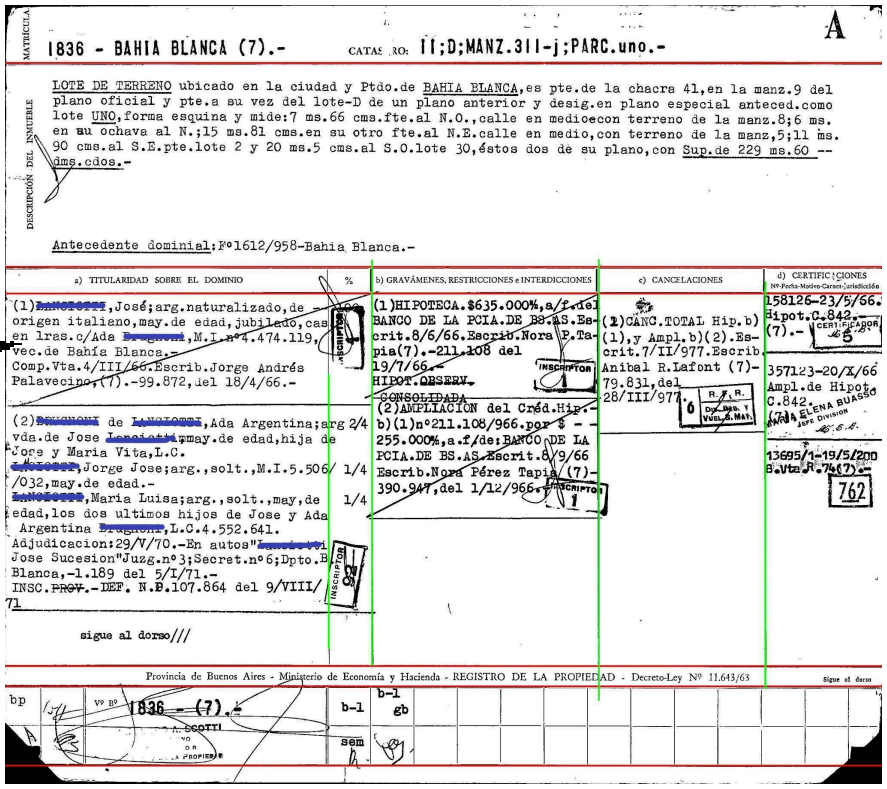


Fig. 13 - Imagen 1836 original superpuesta con la imagen erosionada

Esto lleva a la conclusión que el orden en donde se encuentran los titulares es variable. Lo cual requeriría de alguna operación extra para determinar cuál es el rectángulo buscado. Todo este proceso podría evitarse realizando un post procesamiento del texto reconocido OCR, con las búsquedas de palabras.

Sin embargo, esta forma de resolver el problema de las columnas, a través de dividir la imagen en rectángulos, podría ser un tema para una investigación a futuro.

Herramientas de procesamiento del lenguaje natural (NLP)

Otra estrategia es utilizar herramientas de procesamiento de lenguaje natural. Natural Language Toolkit (NLTK) [29] es un conjunto de librerías y programas para Python que permiten llevar a cabo muchas tareas relacionadas con el Procesamiento del Lenguaje Natural. Dichas tareas están programadas de manera eficiente en NLTK y pueden ser realizadas directamente en la aplicación que se está desarrollando.

Para procesar el texto se debe tokenizar. Este proceso implica separar en palabras el contenido y generar un arreglo con ellas.

```
texto="La casa de Arturo"
tokenized=nltk.word_tokenize(texto)
```

Y luego taggear el texto tokenizado, que lo que hace es formar pares, en donde la primera componente es la palabra y la segunda el tipo de palabra que se detectó:

```
nltk.pos_tag(tokenized)
[('La', 'NNP'), ('casa', 'NN'), ('de', 'FW'), ('Arturo', 'NNP')]
```

La librería detecta varios tipos de palabras. Adverbios, adjetivos, etc, en inglés.

Para la aplicación de búsqueda de titulares abordada en el presente trabajo, son relevantes aquellos pares cuya segunda componente es NNP, siglas que identifican a los nombres propios, el resto serán ignorados. Se probará esto con el texto reconocido de la imagen 1836 de la figura 13, que básicamente taggea las palabras en base a su forma, y en el caso de los nombres propios se fija aquellos que empiecen con mayúsculas. Este método puede no ser exacto, ya que como vemos en el ejemplo existen palabras escritas completamente en mayúsculas, HIPOTECA o AMPLIACION, que son tomadas como sustantivos propios.

Otro punto a tener en cuenta en cuento a la identificación de sustantivos propios es que si bien la normativa (Digesto de Asientos Registrales) determina que el apellido debe ser escrito con mayúsculas en todas sus letras, esto no es suficiente para determinar un apellido, pues muchas veces aparece el apellido de casada (siglo XX) de una persona en la cual la palabra “de” está escrita con minúsculas. Entonces esto estaría actuando como un separador en el split, algo que no es deseable.

Otra regla es la de separar el apellido y el nombre con “,” (el carácter coma) la cual no siempre se cumple. Muchas veces era sustituído por un “;” (punto y coma), o directamente se omitía.

También es sabido que al final del nombre completo colocan el “;”, pero el modelo de Tesseract particularmente tiene una dificultad para determinar este

símbolo. Muchas veces lo reconoce como una “j” (carácter jota), el cual se asemeja mucho.

Lo que sí puede afirmarse en general, es que, después del nombre, aparece la nacionalidad, en forma abreviada. La mayoría de las veces es “arg.”.

Como mencionamos previamente, la librería NLTK al buscar nombres de personas, lo que en realidad hace es buscar palabras que empiecen con mayúsculas. Con la librería RE es posible realizar esta misma acción a través de un patrón de expresión regular adecuado, sin la necesidad de tokenizar la salida del proceso de OCR. Implicaría un paso menos en todo el proceso de post procesamiento. Esta librería fue utilizada finalmente para encontrar los metadatos en las imágenes, a través de palabras claves como Catastro, Antecedentes dominiales, Titularidad, Gravámenes o Cancelaciones.

El método de búsqueda de metadatos, intenta identificar palabras claves dentro de la matrícula que representen datos relevantes para nuestra búsqueda.

Como primer paso debemos identificar la línea que separa la descripción del inmueble con el contenido de los rubros A, B, C y D.

En dicha línea podrían encontrarse las palabras claves:

- Titularidad o TITULARIDAD;
- Gravámenes, GRAVAMENES o GRAVÁMENES
- Interdicciones o INTERDICCIONES
- CANCELACIONES o Cancelaciones
- Restricciones o RESTRICCIONES
- Certificaciones o CERTIFICACIONES

Como se observa, las palabras pueden aparecer completamente en mayúsculas, o empezar con mayúsculas y seguir con minúsculas. Y en caso de estar con mayúsculas, podrían o no tener tilde. Si se encuentra alguna de estas palabras, puede decirse que se ha encontrado un separador, y, por consiguiente, más arriba estará la descripción del inmueble y más abajo aparecerán los titulares. Son palabras que no se usan en otra parte de la matrícula, y se debe tener cuidado si se encuentra más de una, de no volver a separar el texto en dos partes.

Otras de las palabras claves que se pueden encontrar y que son útiles, podrían ser:

- "Catastro" o "CATASTRO": Luego de esta palabra estará la nomenclatura catastral, que es clave para la identificación del inmueble, por parte de la oficina de catastro.
- "antecedentes", "Antecedentes", "ANTECEDENTES", "DOMINIALES", "dominiales", "Dominiales": Son palabras que indican que luego vendrá el antecedente dominial, que es una identificación del inmueble (puede ser otro número de inscripción o un folio) que está relacionado con la matrícula que estamos analizando, pero no la identifica, es decir, no se deberá tener en cuenta, en caso de que se requiera, al momento de una futura búsqueda por número de inmueble.

Otro punto analizado inicialmente fue la definición de stop words que consiste en definir una lista de palabras que no son relevantes para la búsqueda dado que se repiten, y pueden entorpecer el proceso. Algunas de ellas son: `'COPROPIEDAD', 'DECRETO', 'LEY', 'DECRETOLEY', 'BAHIA', 'LOTE', 'TERRENO', 'FEDERAL','ANTECEDENTE', 'PLANO', 'CANCELACIONES', 'DPTO', 'MINISTERIO', 'HACIENDA', 'SIGUE', 'REGISTRO', 'PROVINCIA', 'RESTRICCIONES', 'EDIFICADO', 'UBICADO', 'CIUDAD', 'CALLE', 'APROBADO', 'SOBRE', 'DOMINIO', 'INTERDICCIONES', 'EMPLEADA', 'EMPLEADAS', 'EMPLEADO', 'EMPLEADOS', 'HIJO', 'HIJOS', 'HIJA', 'HIJAS', 'DEFINITIVA', 'BUENOS', 'AIRES', 'DORSO', 'PROPIEDAD'`. Este método no ha sido aplicado finalmente al advertir que algunas de estas palabras pueden ser útiles para delimitar campos dentro de la matrícula (por ejemplo, Cancelaciones) o metadatos.

La estrategia de NLP utilizando la librería de expresiones regulares RE de Python resultó ser la más adecuada para la identificación de las palabras según su ubicación.

El uso de una base de datos documental permite realizar las tareas mencionadas previamente en forma automática y transparente para el usuario final, como se mencionó en el artículo del Thomas et al. Una vez almacenadas todas las

palabras en la base de datos documental, el script para recuperar el nombre de los titulares solo debe preocuparse por identificar correctamente cada uno de ellos. En el siguiente capítulo se presentarán las bases de datos documentales estudiadas en esta tesina y la selección de ElasticSearch para implementar el desarrollo.

Capítulo V. Base de datos documentales

Como ya se dijo, se ha utilizado una base documental o NoSQL (not only Sql) para almacenar el texto extraído de las imágenes. Dentro del ranking mundial de bases de datos [30] se encuentran dos productos relevantes al momento de desarrollar la presente tesina: ElasticSearch y MongoDB. En este capítulo se hará una comparación entre las mismas, así como también con la base de datos relacional MySql, la más utilizada dentro de las opciones de código abierto.

Comparación de las bases de datos documentales

ElasticSearch es un servidor de búsqueda basado en Lucene [31]. El proyecto Apache Lucene liberó una biblioteca de búsqueda central denominada Lucene Core así como también PyLucene, una extensión de Python para facilitar la conexión con Lucene. Lucene Core es una biblioteca de Java que proporciona potentes funciones de indexación y búsqueda de información textual dentro de sus proyectos, así como revisión ortográfica, resaltado de visitas y capacidades avanzadas de análisis/tokenización. El proceso de indexación consiste en analizar y extraer de entre toda la información disponible, la verdaderamente relevante. Posteriormente, con esa información se crea el índice a partir del cual se realizarán las búsquedas. ElasticSearch, en consecuencia, provee un motor de búsqueda de texto completo, distribuido y con capacidad de multitenencia con una interfaz web RESTful y con documentos JSON. También provee lo que se llama Elastic Stack, compuesto por Logstash y Kivana [32]. Esta última es una herramienta que permite la exploración visual y el análisis en tiempo real de los datos en ElasticSearch. Se cuenta también con una versión de ElasticSearch en la nube denominada ElasticCloud [33], que incluye una versión de Kivana y permite una prueba gratuita de 14 días.

MongoDB [34] es una base de datos NoSQL orientada a documentos liberada a mediados de la década de 2000. Se utiliza para almacenar volúmenes masivos de datos. A diferencia de una base de datos relacional SQL tradicional, MongoDB no se basa en tablas y columnas, sino que los datos se almacenan como colecciones y documentos.

MySQL [35] es un sistema de bases de datos de código abierto de Oracle, que se utiliza en todo el mundo para gestionar bases de datos. Se basa en el álgebra relacional y se utiliza principalmente para el almacenamiento de datos de diversas aplicaciones basadas en Web como el desarrollo de la presente tesina. La razón por la cual la incluimos en esta comparación, es que es la base de datos de código abierto más utilizada del mundo.

La tabla 4 muestra un análisis comparativo entre las tres bases de datos, dos de ellas documentales o NoSql, y una relacional, para mostrar las ventajas de unas sobre la otra.

Aspecto	**MongoDB**	**ElasticSearch**	**MySQL**
Tipo de base de datos	NoSQL	NoSQL	Relacional
Escalabilidad	Escalabilidad horizontal fácilmente	Escalabilidad horizontal fácilmente	Escalabilidad vertical y horizontal
Esquema de datos	Esquema flexible y dinámico	Esquema flexible y dinámico	Esquema fijo y estructurado
Búsquedas de texto	No es su enfoque principal, pero admite índices de texto completo	Principalmente diseñado para búsquedas de texto	Admite búsquedas de texto con índices de texto completo
Transac-ciones	Admite transacciones en versiones recientes	No es su enfoque principal	Admite transacciones ACID
Lenguaje de consulta	Utiliza el lenguaje de consulta MongoDB (Mongo Query Language)	Utiliza una API RESTful basada en JSON	Utiliza SQL (Structured Query Language)
Uso típico	Aplicaciones web y móviles, análisis de datos, IoT	Búsqueda y análisis de texto, logística, monitoreo	Aplicaciones empresariales, sistemas transaccionales

Tabla 4 - Tabla comparativa de bases de datos de elaboración propia en base a la documentación oficial

MongoDB es una base de datos, mientras que ElasticSearch es un motor de búsqueda. Mientras que MongoDB tiene que ver con la flexibilidad en la estructura de los datos, ElasticSearch tiende a ser más preciso y estructurado en la forma en que trata y organizan los mismos.

En cuanto al análisis de performance, según la consultora Quarslab [36]: “Esta brecha repentina y enorme en la respuesta de tiempo de MongoDB en ~9 millones de direcciones IP parece tener como causa que el caché cambia de lecturas de RAM a lecturas de disco. Notamos que las E/S de lectura pasaron de ~600/s a ~8000/s, la velocidad de lectura en el disco pasó de ~20 M/s a ~150 M/s y las fallas de página pasaron de picos de ~4000/s a un nivel promedio de ~ 8.000/segundo. En cambio, las medidas en nuestro servidor con 64 GB respaldan este análisis: el sistema nunca se queda sin RAM y, por lo tanto, el tiempo de respuesta continúa aumentando lentamente sin ninguna rampa repentina”. Este comportamiento puede verse en la figura 14, con intervalos de 100.000 documentos, en más de 23 millones de documentos.

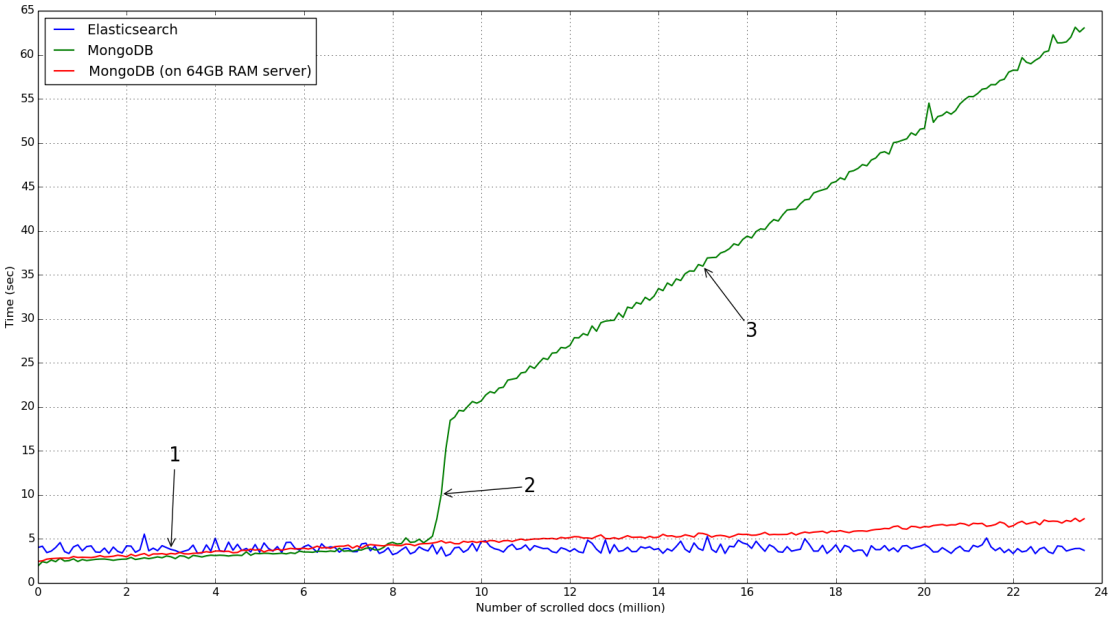


Fig. 14 - Comparación de performance de las bases

Incluso si MongoDB tiene un mejor rendimiento de inserción masiva y es más flexible, ElasticSearch es realmente un motor más prometedor para los requerimientos de este proyecto.

En cuanto a motores de búsqueda, según DB Engines, ElasticSearch ocupa el puesto número uno en los motores de búsqueda y el séptimo en general. MongoDB ocupa el puesto número uno en bases de datos de almacenamiento de documentos y el quinto en general. La figura 15 muestra el ranking de bases de datos según DB Engines a junio de 2024.

Rank			DBMS	Database Model	Score		
Jun 2024	May 2024	Jun 2023			Jun 2024	May 2024	Jun 2023
1.	1.	1.	Oracle	Relational, Multi-model	1244.08	+7.79	+12.61
2.	2.	2.	MySQL	Relational, Multi-model	1061.34	-22.39	-102.59
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	821.56	-2.73	-108.50
4.	4.	4.	PostgreSQL	Relational, Multi-model	636.25	-9.30	+23.43
5.	5.	5.	MongoDB	Document, Multi-model	421.08	-0.58	-4.29
6.	6.	6.	Redis	Key-value, Multi-model	155.94	-1.86	-11.41
7.	7.	↑ 8.	Elasticsearch	Search engine, Multi-model	132.83	-2.52	-10.92
8.	↑ 9.	↑ 11.	Snowflake	Relational	130.36	+9.03	+16.23
9.	↓ 8.	↓ 7.	IBM Db2	Relational, Multi-model	125.90	-2.56	-18.99
10.	10.	10.	SQLite	Relational	111.41	-2.91	-19.81
11.	11.	↓ 9.	Microsoft Access	Relational	101.16	-3.75	-33.29
12.	12.	12.	Cassandra	Wide column, Multi-model	98.83	-3.06	-9.73
13.	13.	13.	MariaDB	Relational, Multi-model	91.04	-2.17	-6.28
14.	14.	14.	Splunk	Search engine	89.10	+2.65	-0.35
15.	15.	↑ 18.	Databricks	Multi-model	81.08	+2.47	+15.27
16.	16.	16.	Microsoft Azure SQL Database	Relational, Multi-model	76.78	-1.20	-2.18
17.	17.	↓ 15.	Amazon DynamoDB	Multi-model	74.45	+0.38	-5.46
18.	18.	↓ 17.	Hive	Relational	59.76	-1.42	-15.76
19.	19.	↑ 20.	Google BigQuery	Relational	58.10	-2.28	+3.46
20.	20.	↑ 21.	FileMaker	Relational	47.91	-0.29	-6.47
21.	↑ 23.	↑ 22.	Neo4j	Graph	44.89	+0.44	-7.87

Fig. 15 - Ranking mundial de bases de datos

Por lo antes expuesto en cuanto a performance y funcionalidades ofrecidas, ElasticSearch es el motor de búsqueda seleccionado para el presente proyecto.

Operaciones sobre el índice ElasticSearch

Recuperando información del índice ElasticSearch

En el contexto de ElasticSearch, el campo o atributo "`_score`" en la respuesta JSON se refiere al puntaje de relevancia asignado a cada documento, en función de la consulta realizada. Elastic utiliza el modelo de espacio vectorial y el algoritmo TF-IDF (Term Frequency-Inverse Document Frequency o frecuencia de términos y la frecuencia de documentos inversa) para calcular este puntaje.

El puntaje de relevancia (`_score`) indica qué tanto coincide un documento con los términos de búsqueda y la consulta realizada. Cuanto mayor sea el puntaje,

más relevante se considera el documento en relación con la consulta. Los documentos con puntajes más altos se clasifican más arriba en los resultados de búsqueda. Es importante tener en cuenta que este puntaje de relevancia (`_score`) es relativo y solo se utiliza para comparar la relevancia de los documentos dentro de una consulta específica. Los puntajes no tienen una escala o significado absoluto en sí mismos y no se pueden comparar entre consultas diferentes.

El `_score` es útil para ordenar los resultados de búsqueda de ElasticSearch de manera que los documentos más relevantes aparezcan en los primeros lugares de la lista. A continuación, se muestra un ejemplo de cómo funciona este atributo a través del Dev Tools de ElasticSearch (herramientas de desarrollador), que utiliza un lenguaje de consultas basado en JSON:

```
GET titulares/_search
{
  "query":{
    "bool": {
      "must": [
          {"match": {
            "nombre":"Pedro"
          }}
      ],
      "should": [
          {"match": {
            "nombre":"Pedro"
          }}
      ]
    }
```

La consulta anterior dará como resultado todos aquellos documentos que contengan Pedro en su nombre, como se muestra a continuación:

```
{
  "took": 566,
  "timed_out": false,
```

```
"_shards": {
  "total": 1,
  "successful": 1,
  "skipped": 0,
  "failed": 0
},
"hits": {
  "total": {
    "value": 3,
    "relation": "eq"
  },
  "max_score": 1.3551913,
  "hits": [
    {
      "_index": "titulares",
      "_id": "xdzCoYIBt4WkyusFKaAg",
      "_score": 1.3551913,
      "_source": {
        "nroinscri": 32,
        "archivo": "img1.tif",
        "nombre": "Pedro",
        "apellido": "Gonzalez"
      }
    },
    {
      "_index": "titulares",
      "_id": "xtzEoYIBt4WkyusFb6BS",
      "_score": 1.0779929,
      "_source": {
        "nroinscri": 312,
        "archivo": "imag1co.tif",
        "nombre": "Pedro ALberto",
        "apellido": "Ramirez"
      }
```

```
      },
      {
        "_index": "titulares",
        "_id": "dbLNoYIBnWTqGjjtjg7I",
        "_score": 0.8949375,
        "_source": {
          "nroinscri": 123,
          "archivo": "hola1234.jpg",
          "nombre": "Alberto Pedro Perez"
        }
      }
    ]
  }
}
```

Es de destacar que no permite una búsqueda puntual, ya que se utiliza para búsquedas masivas, al estilo de los buscadores Web.

En la aplicación a desarrollar las consultas son simples, sólo se deberá buscar coincidencias entre las palabras y retornará información con coincidencias parciales. Como consecuencia, se optará por las primeras respuestas, o utilizar el criterio personal para aceptar o descartar las opciones.

Actualizando el índice ElasticSearch

A continuación, se muestran dos ejemplos de cómo agregar elementos al índice. En el primer caso se especifican valores para los campos nroinscri, archivo y nombre mientras que para el último caso se especifican valores para nroinscri, archivo, nombre y apellido.

```
POST titulares/_doc
{
  "nroinscri": 123,
  "archivo" :"hola1234.jpg",
  "nombre": "Alberto Pedro Perez"
```

```
  }
POST titulares/_doc
{
  "nroincri": 312,
  "archivo" :"img1co.tif",
  "nombre": "Pedro ALberto",
  "apellido": "Ramirez"
 }
```

Podemos observar que pueden omitirse algunos datos y aun así agregar el registro al índice, ya que las bases documentales no se limitan al uso de estructuras rígidas. Además, la gran ventaja de ElasticSearch es que, si se tiene el número de inscripción, se podría utilizar para el acceso al registro buscado. De igual forma, si sólo se tiene el nombre del archivo correspondiente a la imagen buscada.

En el presente trabajo se ha optado por almacenar el texto completo del OCR en el índice ElasticSearch, a pesar de tener muchos caracteres, en principio, no relevantes. Esta decisión llevará a perder la menor cantidad de información posible, y con herramientas de procesamiento de lenguaje natural (post procesamiento) seleccionaremos la información relevante. Lorena Talamé, Alejandra Cardoso y Matías Amor [37], en el trabajo llamado “Comparación de herramientas de procesamiento de textos en español extraídos de una red social para Python”, analizan cinco módulos o librerías para el lenguaje Python, con el objetivo de comparar algunas funciones básicas del procesamiento de textos, aplicadas a textos cortos.

Capítulo VI. Arquitectura propuesta

Basándose en el patrón Modelo-Vista-Controlador (MVC), se necesitan interfaces que se comuniquen con nuestro modelo. Debido a que se ha optado por Python como lenguaje de programación para desarrollar el diseño de la aplicación de esta tesina, se debe buscar un framework [38] para manejar la información. Un framework es un marco de trabajo o estructura que se utiliza para desarrollar software. Un framework es por tanto un conjunto de herramientas y módulos que pueden ser utilizados para varios proyectos. En el mercado se presentan dos opciones principales: Django y Flask.

Comparación entre frameworks de MVC de Python

Django [39] sigue el patrón de diseño Modelo-Vista-Template (MVT), que es una variante del patrón Modelo-Vista-Controlador (MVC). Este framework sigue la filosofía "baterías incluidas", que abarcan desde la gestión de bases de datos y la autenticación de usuarios hasta un potente sistema de administración, sin tener que depender de librerías externas. Django sigue una filosofía proactiva de seguridad. Incluye características de seguridad integradas, como protecciones contra ataques comunes (inyección de SQL, CSRF, XSS) y promueve buenas prácticas de seguridad por defecto.

Flask [40] es un “micro” framework. Se enfoca en proporcionar lo mínimo necesario para poner a funcionar una aplicación básica en cuestión de minutos. Si se requiere más funcionalidades es posible extenderlo con las Flask Extensions [41]. Este framework incluye un servidor web de desarrollo para probar las aplicaciones sin tener que instalar uno. También trae un depurador y soporte integrado para pruebas unitarias. Es compatible con Python3, por lo tanto, es posible utilizar la codificación de caracteres Unicode, y es 100% compatible con el estándar WSGI [42]. Con el uso de un decorador Python se puede hacer una aplicación con URL simples y limpias. Soporta el uso de cookies seguras y el uso de sesiones. También se apoya en el motor de plantillas Jinja2, que permite, de forma sencilla, renderizar vistas y respuestas. No tiene ORM, wrappers o configuraciones complejas, eso lo convierte en un candidato ideal para aplicaciones ágiles o que no necesiten manejar ninguna dependencia. Si se necesita trabajar

con base de datos sólo se debe utilizar una extensión. Su código es Open Source y está amparado bajo una licencia BSD. Se puede ver su código en GitHub.

La tabla 5 muestra una tabla comparativa entre ambas tecnologías.

Aspecto	Flask	Django
Complejidad	Microframework ligero y minimalista	Framework complejo de alto nivel
Flexibilidad	Altamente flexible, permite elegir las herramientas	Ofrece una estructura más rígida con "baterías incluidas"
Escalabilidad	Escalable, la estructura depende de las decisiones del desarrollador	Facilita la escalabilidad de proyectos grandes y complejos
Aprendizaje	Fácil de aprender, adecuado para principiantes	Mayor curva de aprendizaje debido a sus convenciones y estructura
Casos de uso	Ideal para proyectos pequeños o medianos, prototipos, alta personalización	Ideal para proyectos grandes y complejos, aplicaciones empresariales, comercio electrónico, aplicaciones con convenciones establecidas

Tabla 5 - Comparación de diseño propio en base a la documentación oficial

Debido a la complejidad y tamaño del desarrollo necesario para esta tesina, que es entre pequeño y mediano, y la curva de aprendizaje necesaria para incorporar cualquiera de los dos frameworks, se ha optado por Flask. Recientemente se han incorporado otros frameworks para Web de Python, como son Pyramid y Web2py, pero la comparación ha sido entre los que hemos conocido en nuestra Facultad.

Aplicación web consumidora

La función de la aplicación web consumidora es comunicarse con el servidor Flask para obtener la información almacenada en ElasticSearch, incluyendo las imágenes procesadas. Esta aplicación web, desarrollada con Flask y Bootstrap, se encarga de mostrar las imágenes y otros datos relevantes en el navegador al usuario interno que recibe la solicitud de un usuario externo. Aquí es donde se

aplicarían los estilos de Bootstrap para mejorar la apariencia y la experiencia del usuario.

Como se mencionó previamente, en lugar de una base de datos relacional o NoSQL tradicional, se utiliza ElasticSearch como sistema de almacenamiento, y servirá para el caso de esta tesina que es de búsqueda de textos. Elasticsearch se integra bien con Flask. En el siguiente capítulo se presentará el flujo necesario para obtener los resultados esperados.

Aplicación para el procesamiento de imágenes

Esta aplicación se ejecuta de forma independiente en el servidor y se encarga de procesar las imágenes, alimentar con datos a Elasticsearch y realizar la indexación correspondiente. La aplicación de procesamiento de imágenes puede ser desarrollada utilizando tecnologías y herramientas específicas para el procesamiento de las mismas en Python.

Para que este circuito funcione, se deben realizar varias tareas. La primera en el tiempo, es la digitalización de las imágenes, por parte de un empleado de área (usuario interno), por ejemplo, que escanee una por una las matrículas, de ambos lados, y vaya ingresando el correspondiente número de inscripción de cada una.

En una segunda instancia, un administrador de sistemas deberá hacer correr el servidor de ElasticSearch, en el equipo que se disponga para tal fin. Luego, el mismo operador, deberá ejecutar el software necesario para la extracción de texto, indicando la ubicación de las imágenes al mismo, y el modelo de Tesseract que va a utilizarse para el reconocimiento de caracteres.

Luego un administrador de sistemas debe ejecutar el servidor de Flask, para que la aplicación pueda atender los requerimientos web.

Posteriormente, un usuario externo registrado, hará una solicitud que luego el usuario interno deberá procesar, seleccionándola de una lista de pedidos. Por último, este usuario interno hará una devolución con la respuesta según su calificación, a través de un correo electrónico proporcionado para tal fin.

La figura 15 muestra un esquema con el flujo de información y los agentes que intervienen.

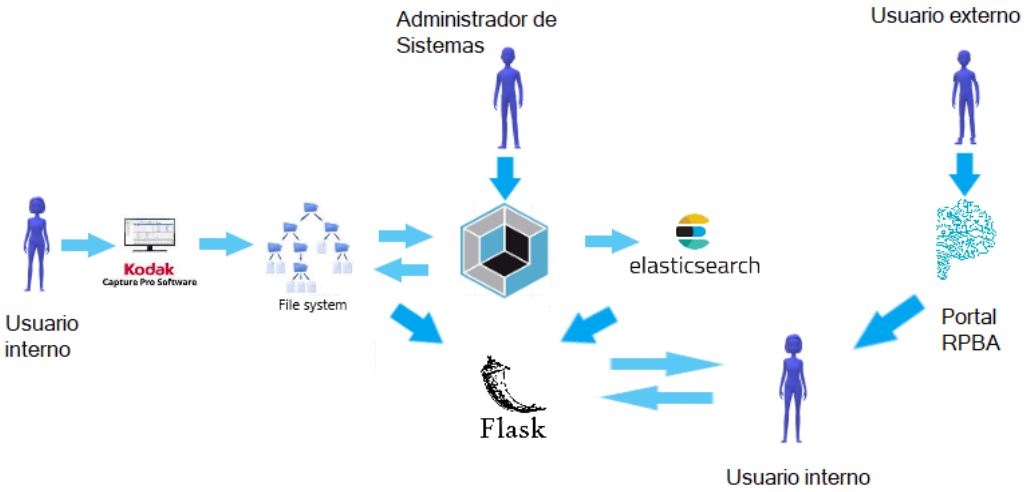


Fig. 15 - Arquitectura propuesta

Capítulo VII. Funcionamiento de la aplicación

En este capítulo se desarrolla cómo será el procedimiento para generar la solicitud del trámite de consulta al historial de titulares.

Este trámite, en principio, estará dentro de lo que se enmarca como publicidad registral, dado que no realiza operaciones que alteren la información dentro de las matrículas, como son por ejemplo una compra-venta de inmueble (documento notarial), o un embargo judicial (generado por un oficio proveniente de algún juzgado). En otras palabras, a través de nuestra consulta, sólo se publicará información contenida dentro del Registro de la Propiedad, pero no podremos alterar dicha información. Como se mencionó en el capítulo II de elicitación de requerimientos, se identifican los siguientes tipos de usuarios: usuario externo, usuario interno y usuario administrador de Sistemas.

Generación de la solicitud

Para diseñar esta solicitud se ha tomado como prototipo, a pedido de la asesora profesional, a la Consulta al Índice de Titulares. Según el instructivo de la consulta al Índice de Titulares:

“Mediante este instructivo se describe el procedimiento para que el público general, que cuente con interés legítimo, pueda acceder, gestionar y recibir el servicio de consulta al Índice de titulares – Créditos Hipotecarios, por internet y con firma digital. Los usuarios profesionales podrán acceder al mismo servicio únicamente desde su cuenta de usuario suscripto, en la ventanilla virtual del organismo. Los trámites ingresados, tendrán código, número y fecha de entrada, durante el horario de la mesa de entradas, es decir de lunes a viernes desde las 8 horas a las 13.30, fuera de ese rango se podrá iniciar el trámite, el sistema le va a informar un número de operación quedando pendiente de su efectivo ingreso hasta el siguiente día y hora hábil. Para acreditar la identidad, luego de realizar el trámite deberá enviar por mail su DNI escaneado a acreditacionTitulares@rpba.gov.ar.”

Generar el pedido

En primer lugar, como se muestra en la figura 16, se debe ingresar a la web del RPBA: www.rpba.gov.ar. Luego, en el menú de la derecha, se busca el enlace “Aplicaciones web”.

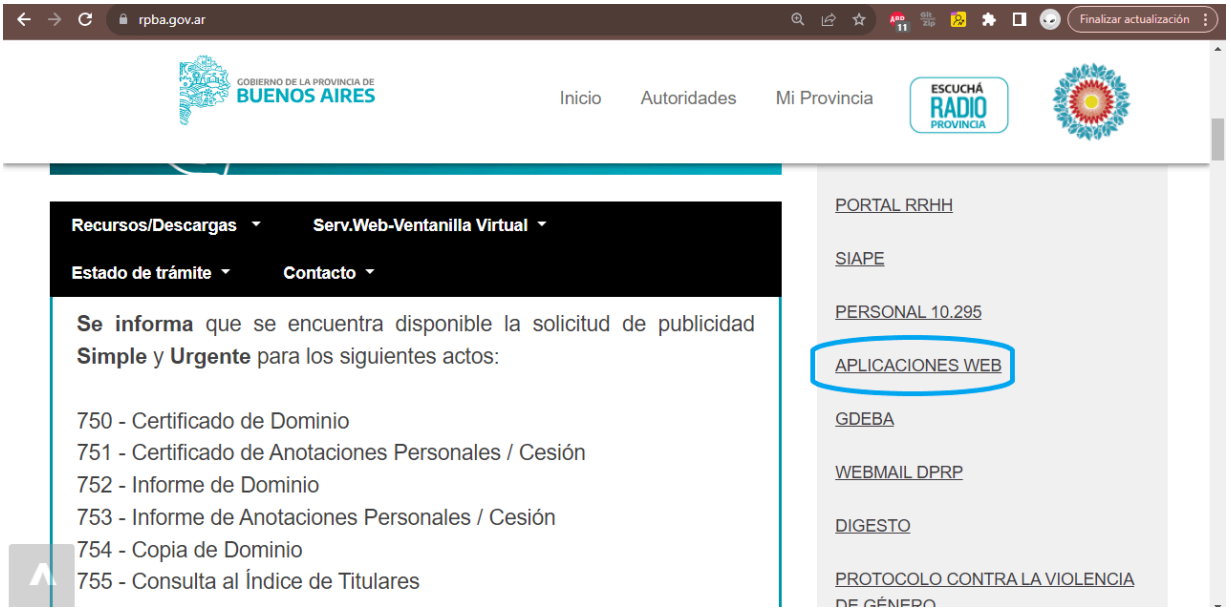


Fig. 16 - Portal RPBA

La figura 17 visualiza la página de identificación de un usuario externo o suscripto.

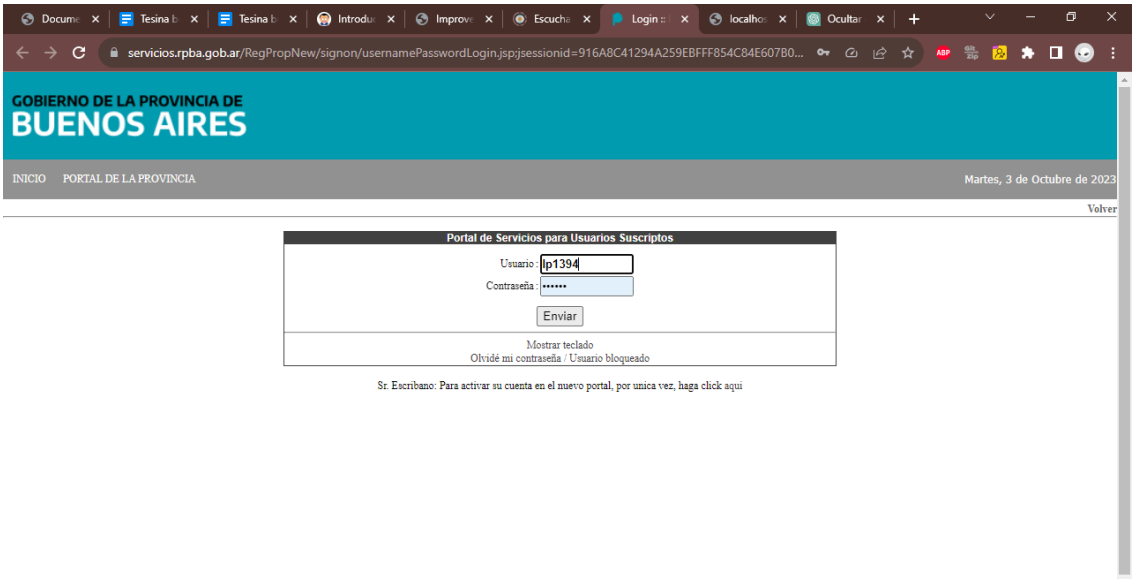


Fig. 17 - Página de login

En el listado de aplicaciones que aparece se debe elegir la opción “Consulta histórico de titulares”, como muestra la figura 18.

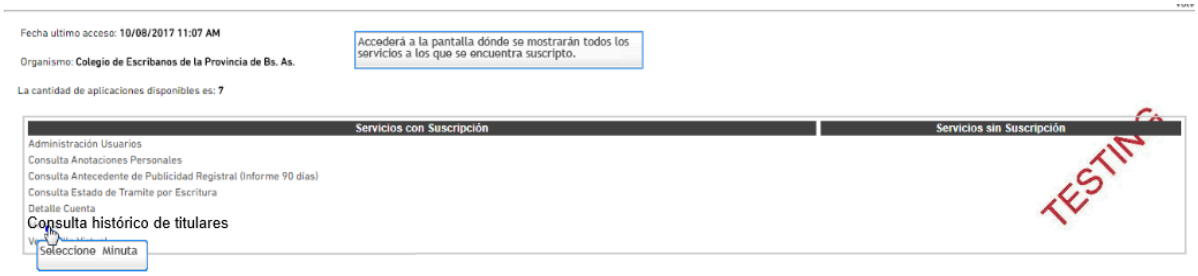


Fig. 18 - Menú de aplicaciones

Luego se llega a un formulario (figura 19) donde deberá completarse el nombre y apellido del titular que se quiere buscar, el correo electrónico donde se recibirá la respuesta, y el timbrado que se utilizará para pagar el trámite, obtenido en la oficina del Colegio de Escribanos dentro del RPBA, generando el pago virtualmente a través del usuario del mismo Colegio, o a través de AFIP.

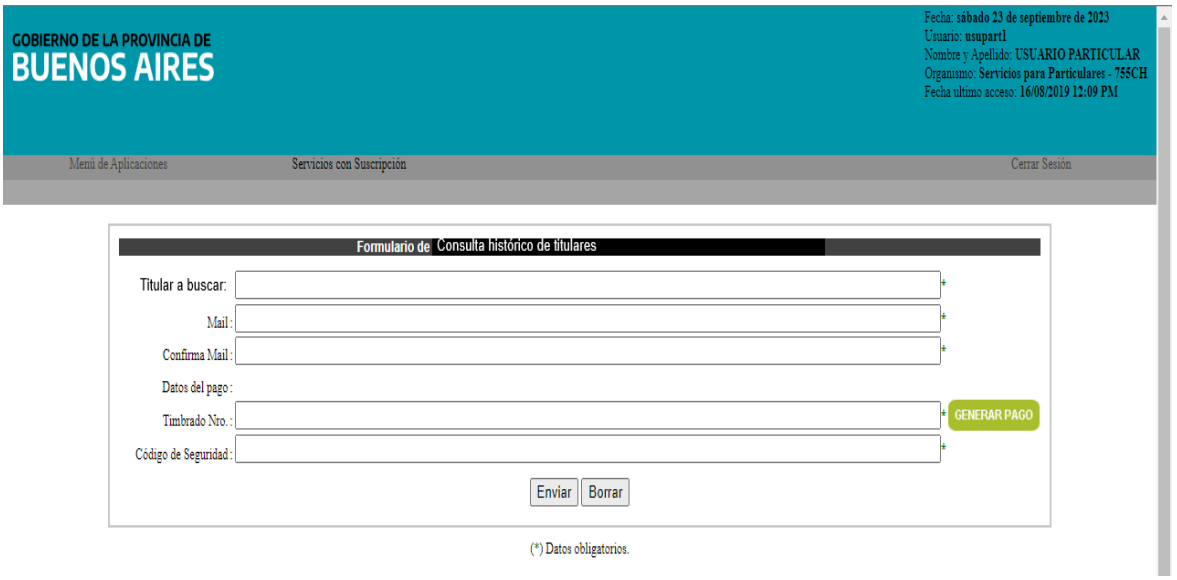
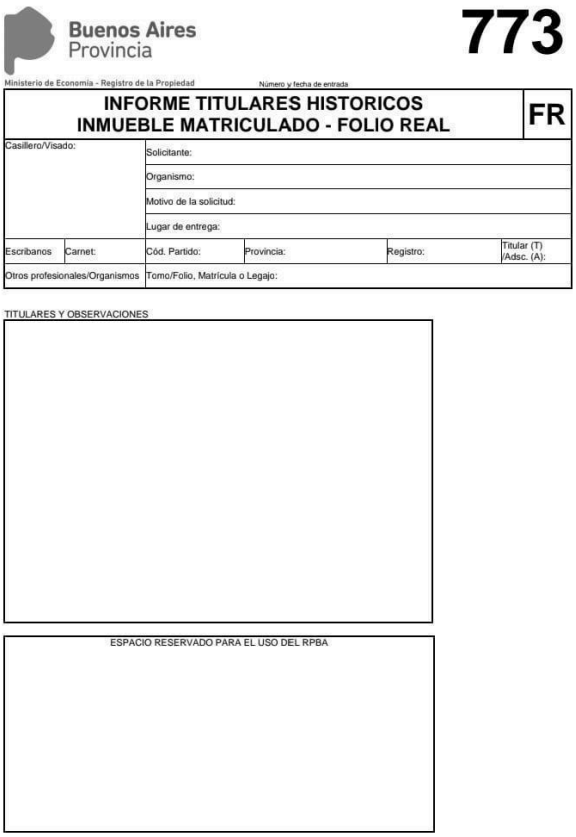


Fig. 19 - Formulario de consulta

Procesar la solicitud

Desde un sistema desarrollado en lenguaje Natural/Adabas de Software AG [43], herramienta ad hoc propia del Registro de la Propiedad, se emitirán los formularios con los pedidos de informes que han hecho los usuarios externos.

La figura 20 muestra el formulario que le llegará al usuario interno.



Buenos Aires Provincia

773

Ministerio de Economía - Registro de la Propiedad

Número y fecha de entrada

INFORME TITULARES HISTORICOS
INMUEBLE MATRICULADO - FOLIO REAL

FR

Casillero/Visado:	Solicitante:				
	Organismo:				
	Motivo de la solicitud:				
	Lugar de entrega:				
Escribanos / Carnet:	Cód. Partido:	Provincia:	Registro:	Titular (T) /Adsc. (A):	
Otros profesionales/Organismos	Tomo/Folio, Matrícula o Legajo:				

TITULARES Y OBSERVACIONES

ESPACIO RESERVADO PARA EL USO DEL RPBA

Fig. 20 - Formulario de solicitud emitido por Natural

Luego de elegir una solicitud, el usuario irá a la aplicación desarrollada que se muestra en la figura 21.

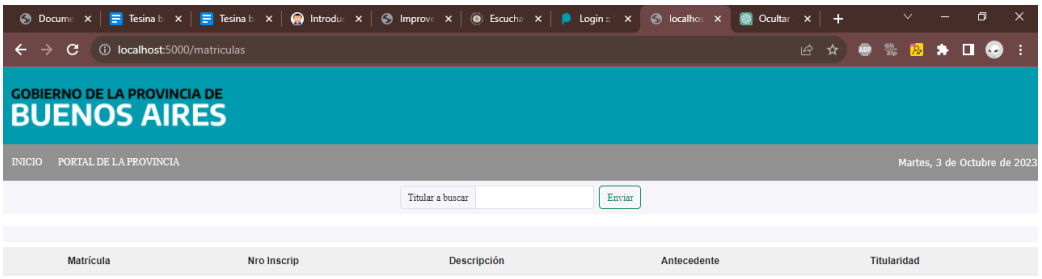


Fig. 21 - Página de procesamiento de la consulta

Se debe ingresar la cadena a buscar (en nuestro caso K**n Alberto José donde se les han ocultado algunos caracteres para proteger la privacidad de la información) y se deberá enviar lo ingresado, como se muestra en la figura 22.

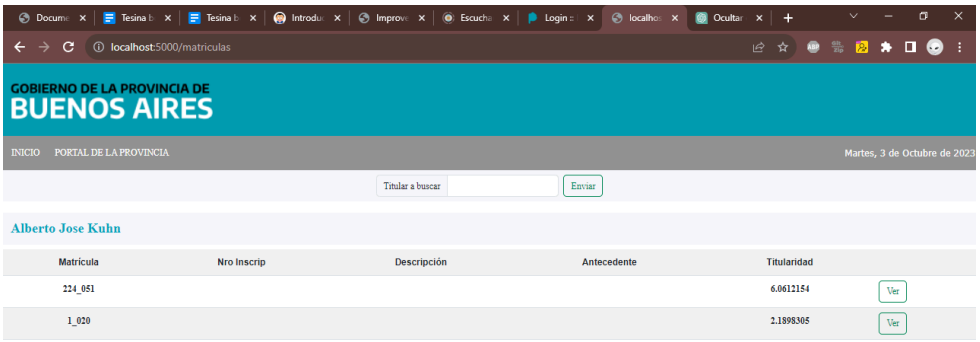


Fig. 22 - Resultado de la consulta

Se deberán observar los resultados, y consultar aquellos que tienen un score alto, es decir, son relevantes a la búsqueda realizada. Es recomendable que sea mayor que 1.5, según la experiencia presentada.

En el ejemplo de la figura 22, se deberá ingresar a “ver” en la matrícula 1_020, que corresponde a la matrícula 1 del partido 20.

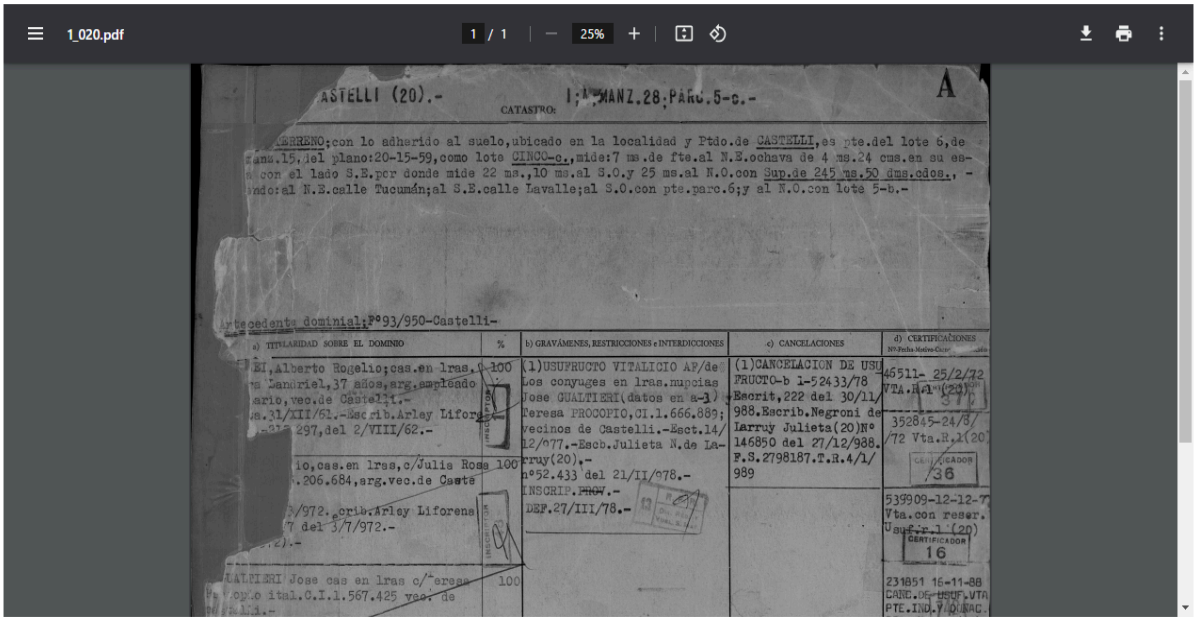


Fig. 23 - Visualización del archivo PDF de la matrícula (20) 1

Si se observa la imagen de la figura 23, se podrá leer “Alberto” y en otro lugar la palabra “José”, pero ningún titular tiene como apellido K**n. Por lo cual esta matrícula no se deberá incluir en la respuesta.

Si en el listado se accede a la matrícula 224_051, es decir, la inscripción 224, del partido 51..

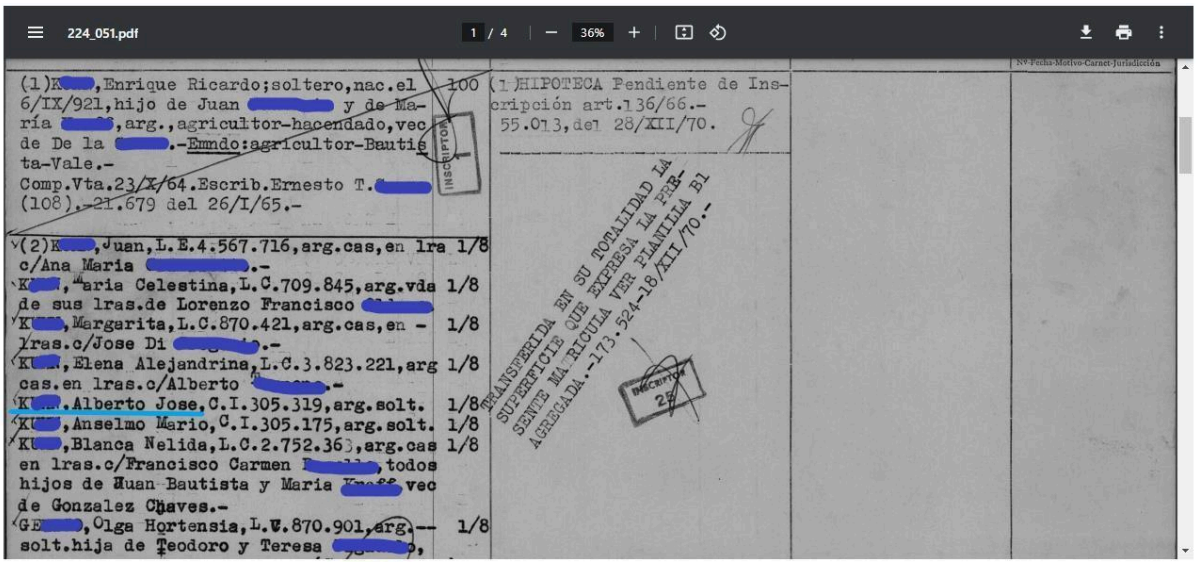


Fig. 24 - Archivo PDF de la matrícula (51) 224

Como se observa en la figura 24, se encuentra “Alberto José K**n” en la columna de titulares, por lo cual, esta inscripción deberá ser incluida en la respuesta que se dará al usuario externo, en forma de PDF.

Resultados y Análisis de las Pruebas de Usuarios del RPBA

La retroalimentación con los usuarios puede ayudar a validar el propósito del proyecto y mejorar su usabilidad. Este apartado se centra en presentar los resultados obtenidos durante la prueba con usuarios, así como en analizar y discutir estos resultados en detalle.

Se han elegido cinco usuarios que desempeñan su función dentro del Área de Registración y Publicidad. Dos de ellos pertenecientes al sector Certificados, uno al sector de Inscripciones, dos abogados pertenecientes al departamento Jurídico, y uno del Sector Distribución, que es el fichero que contiene las matrículas.

Los siguientes nombres han sido proporcionados para la búsqueda (personas humanas y jurídicas):

- K**n, Alberto José
- Av**za, Humberto Ismael
- T**o de Fonte, Liliana
- N**tor H. Fernandez Sociedad de Responsabilidad Limitada
- Bi**chi Oscar Alfredo
- Car**illo Domingo Salvador

A los usuarios se les explicó el proceso de búsqueda y la forma de emisión de una respuesta. Las calificaciones de la herramienta, así como su facilidad de uso ha ido entre 8 y 10, en una escala del 1 al 10.

Una recomendación para mejorar la aplicación ha sido una sugerencia con respecto a la respuesta al usuario final: que sea a través de un botón de enviar, y no redactando un correo electrónico. Otra recomendación es que en el archivo PDF se indique dónde está el titular que se está buscando, ya que la matrícula podría tener varias páginas, y sería un proceso engorroso la localización del mismo. Esto podría resolverse indicando la página en la cual ha sido encontrado el titular, que es un dato que tenemos en nuestra base, recolectado al momento del proceso de OCR.

Entre las utilidades que los usuarios internos le han encontrado al software aparecen:

- La posibilidad de buscar otros campos que no sean el titular, por ejemplo, la nomenclatura catastral, que permite localizar dominios por otra identificación que no sea su número de inscripción.
- La búsqueda de propiedades de personas desaparecidas o que han sido declaradas insanas mentalmente, a las cuales se les ha sustraído su titularidad.
- La ventaja de poder operar la aplicación de manera remota (teletrabajo).
- Verificación para el índice de titulares - acto 755 [44].
- Actualización de la base de datos de titulares.

Capítulo VIII. Conclusiones

El trabajo realizado se enmarca dentro de la Dirección del Registro de la Propiedad Inmueble de la provincia de Buenos Aires, donde desempeño mi labor diaria.

A partir de entrevistas a la coordinadora del proyecto de digitalización, y asesora profesional de esta tesina, surge la necesidad de una herramienta para complementar un proyecto existente, relacionado con la digitalización de documentos en ámbitos gubernamentales.

Esta necesidad propició la investigación sobre soluciones propuestas a nivel nacional e internacional, surgiendo la librería Tesseract como la mejor opción para el reconocimiento de texto, en el momento de desarrollo del presente documento, al poseer modelos entrenados en español antiguo (Spa_old), soportar el entrenamiento con fuentes como Old Typewriter de TrueType y estar basado en redes neuronales LSTM de amplio uso en la actualidad por su posibilidad de entrenamiento.

Asimismo, se investigaron distintas opciones para gestión de este tipo de información, entre ellas, ElasticSearch, MongoDB y MySQL, resultando la primera la más adecuada para indexación basada en texto.

Una vez elegida la librería de reconocimiento de texto basada en Python, y la base de datos documental se procedió a testear distintos modelos con distintas librerías y folios reales cartulares (matrículas) provistos por la asesora profesional.

Luego, se diseñó una aplicación basada en web que permita dar respuesta a los requerimientos relevados, utilizando historias de usuarios. En forma resumida, constaría de una solicitud, una búsqueda o procesamiento y una confección de un informe como respuesta.

El desarrollo de la aplicación fue realizado utilizando Flask para Python, luego de compararlo con otras opciones, dada su versatilidad y facilidad de uso.

Finalmente, se realizó un test con usuarios, que me permitió construir una ayuda en línea, así como también obtener un feedback real del uso de la aplicación.

Durante la realización del proyecto surgieron o se potenciaron librerías como PaddleOCR o TensorFlow, de las cuales es interesante continuar observando su

evolución. En la mayoría de los casos, entrenar una red convolucional desde cero requiere mucho tiempo y grandes conjuntos de datos. Esto sugiere que en un trabajo de mayor duración, la utilización de estas librerías, para el caso de máquinas de escribir antiguas, podría traer mejores resultados en el proceso de OCR, lo cual sería material para otro trabajo.

La tecnología avanza, así como también tareas repetitivas buscan resolverse utilizando IA. Es relevante continuar indagando el impacto de la misma en este tipo de soluciones.

Respecto a la gestión de solicitudes de informes de dominio, sería oportuno como trabajo futuro sustituir la impresión del formulario 773, presentado en el capítulo VII, de modo que el usuario interno que procesa la solicitud, visualice un listado con los pedidos pendientes y seleccione uno de ellos sin necesidad de transcribir el nombre de la persona sobre la que se está realizando la búsqueda.

El código del proyecto se encuentra disponible en https://github.com/sebajoray/DIGITALIZACION_MATRICULAS/tree/master.

Anexo 1

Script de detección de grillas dentro de la imagen

```
def GetGridFromImage(image, maxLineGap):
# Kernel para erosionar (y luego dilatar) de altura de un caracter.
    kernel = np.ones((maxLineGap,1), np.uint8)
# aplica kernel dejando uniones de pixeles con longitud mayor a la de un caracter
# esto deja lineas horizontales si la imagen esta alineada a los margenes
    imageV = cv2.erode(imageBin, kernel)
    imageV = cv2.dilate(imageV, kernel, iterations=2)
#aplica kernel transpuesto, solo deja lineas horizontales
    imageH = cv2.erode(imageBin, kernel.T)
    imageH = cv2.dilate(imageH, kernel.T, iterations=2)
      # combina las lineas horizontales y verticales. Intenta dejar solo la grilla
    imageOut = cv2.bitwise_or(imageV, imageH)
    return imageOut
#Retorna una imagen solo con lineas horizontales y verticales
# Parametros:
#   - una imagen blanco y negro con lineas verticales y horizontales
#     - el largo minimo que debe tener una linea horizontal para ser considerada
#   - el largo minimo que debe tener una linea  vertical para ser considerada
#   - un tamaño (en pixeles) para considerar cuando una linea cortada es continua
#   - el alto promedio de un caracter
```

```
    # Retorna:
    #   - Tupla con lineas horizontales y verticales
    def GetLinesFromImage(image, minLineWidth, minLineHeight
,maxLineGap, charAvgHeight):
        (imgH, imgW) = image.shape

        minLineLength = min(minLineWidth, minLineHeight)
         lines = cv2.HoughLinesP(image, rho=1, theta=np.pi /
2,         threshold=150,         minLineLength=minLineLength,
maxLineGap=maxLineGap)

        lh = list()
        lv = list()
        if lines is not None:
            for i in range(0, len(lines)):
                    # separa las lineas en horizontales y
verticales
                (x1,y1, x2,y2) = lines[i][0]
                lx = abs(x2-x1)
                ly = abs(y2-y1)
                if (lx > ly):
                        if lx > minLineWidth: # verifica que
cumplan con un minimo largo
                            # no agrega lineas sobre bordes
horizontales a distancia de un caracter
                             if y1 > charAvgHeight and y2 <
imgH-charAvgHeight:
                            # agrega coordenadas de la linea
con su longitud
                           lh.append([x1, y1, x2, y2, lx])
                else:
                        if ly > minLineHeight: # verifica que
cumplan con un minimo largo
```

```
                                # no agrega lineas sobre bordes
horizontales a distancia de un caracter
                                     # las coordenadas x estan
invertidas!
                                 if x2 > charAvgHeight and x1 <
imgW-charAvgHeight:
                                # agrega coordenadas de la linea
con su longitud
                               lv.append([x2, y2, x1, y1, ly])

              # ordena las lineas horizontales de arriba hacia
abajo
               lh.sort(key=lambda line: line[1]) # ordena por
cordenada y1 (arriba a abajo)
               lv.sort(key=lambda line: line[0]) # ordena por
cordenada x1 (derecha a izquierda)

                #filtra lineas duplicadas (son consecutivas a
nivel pixel pero son conceptualmente la misma)
             listH = [ [0, lh[0][1], imgW, lh[0][1], imgW] ]
             for i in range(1, len(lh)):
                 (antLn, actLn) = (listH[-1], lh[i])
                      if actLn[1]-antLn[1] < charAvgHeight: #
menos distancia que 1 caracter de alto?
                       listH[-1] = [0,actLn[1], imgW ,actLn[3],
imgW]
                  else:
                                listH.append([0,actLn[1],  imgW
,actLn[3], imgW])

                #filtra lineas duplicadas (son consecutivas a
nivel pixel pero son conceptualmente la misma)
             listV = [lv[0]]
             for i in range(1, len(lv)):
```

```
                (antLn, actLn) = (listV[-1], lv[i])
                    if actLn[0]-antLn[0] < charAvgHeight: #
menos distancia que 1 caracter de alto?
                        listV[-1] = [actLn[0], min(antLn[1],
actLn[1]),actLn[2],max(antLn[3], actLn[3]),actLn[3]]
                                                listV[-1].append(
listV[-1][3]-listV[-1][1] )
                else:
                    listV.append(actLn)

        return (listH, listV)

    # path
    path = r'1-1-1-1838-.tif'

    # Reading an image in default mode
    image = cv2.imread(path)

    (imgH, imgW, _) = image.shape

    # calcula algunos valores en funcion del tamaño de la
image
    minLineWidth  = int(imgW*0.50) # 50% del ancho
    minLineHeight = int(imgW*0.30) # 35% del alto
    maxLineGap    = int(imgW*0.08) # 10% del ancho
    charAvgHeight = 32 #alto promedio de un caracter

    imageGray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)

    # convierte la imagen en blanco y negro  y luego
invierte para poder aplicar
    # operaciones morfologicas como erosion. Queda fondo
negro y lineas blancas
```

```
    (T, imageBin) = cv2.threshold(imageGray, 128, 255,
cv2.THRESH_BINARY_INV)

    # obtiene lineas horizontales y verticales (puede haber
basura)
    imageBinGrid = GetGridFromImage(imageBin, maxLineGap)

    # recupera lineas de referencia (mas largas) que sirven
para caracterizar la planilla
    (linesH, linesV) = GetLinesFromImage(imageBinGrid,
minLineWidth, minLineHeight, maxLineGap, charAvgHeight)

    for l in linesH:
            cv2.line(image, (l[0], l[1]), (l[2], l[3]),
(0,0,255), 3, cv2.LINE_AA)
        print("LH: (%d,%d)->(%d)" % (l[0], l[1], l[4]) )

    for l in linesV:
            cv2.line(image, (l[0], l[1]), (l[2], l[3]),
(0,255,0), 3, cv2.LINE_AA)
        print("LV: (%d,%d)->(%d)" % (l[0], l[1], l[4]) )
    # invierte la imagen para obtener el fondo blanco y
lineas negras
    imageOut = 255-image
    # Displaying the image
    #cv2.imshow(window_name, image)
    plt.imshow(image)
    cv2.imwrite(path+'-BigLinesGrid.png', imageBinGrid)
    cv2.imwrite(path+'-BigLines.png', image)
```

Anexo 2

A continuación se presentan los resultados de las pruebas con las diferentes librerías sobre la imagen de la figura 13, correspondiente a la matrícula 1836 de Bahía Blanca.

1 Keras

El proceso arrojó el siguiente resultado:

s itidmanzzilei sparcounos blanca czlr bahia 1836 catas ro lote de terreno ubicado 1a ciudad ptdonde en bahia blancay ptes 1a y de chacra alzen es 1a manzsg del plano oficial ptera del loted de plano y sx vez anterior desige e an plano especial y en anteceds como lote unos forma 66 esquina midett fter al ne oes calle y mss cms medi oecon terreno de la manzaik6 en mse 81 ochava al nail5 en au fter mse cms otro al nees calle en su diog terreno en me con de la manzy5111 msr 90 all ses 2 pteslote 20 cmse mss5 cmsal scoalote estos 300 dos de y planoy 229 sups de 60 su con msr site e cmss cdoss e antece dente domini f16121ssbahia al blancas ts certific di ciones gravamenes titularidad sobre el dominio 0 by restricciones interdicciones cancelaciones c na fccha carncturisdiccion motivo 5812131766 1 lyianci jhipotecas srse oltigjose nat uralizadon de i argo ipotcons pciade uicanctotat banco de bsoases hipeby origen la italianos mayade edady jubiledo cas abor cery flc e 171 s critasgtggesctinor jo pe lac lrass clada bc 22 en bru***niy miniotazaal1n llsy ampls sesou pialzlm2l10x08 critsze iittzescrib de bahia del blancas vecs 191116665 conpataensrions anibal laf c7m andres dnt re 35723201x166 inscenton cinoatt palave hipors observn 9948722 1841666 8314 dei 796 del ampl de abuissto r 281 iiifgti ro conspliacion cue42e elena 2 del credshipe 0 vees division nkia maf cebru gn oni anciottig de ada tinai 2j4 11n0211106 966epop jefe argen b arg 54 as jose vaa de lanciottinmayade edadghija ooohga banco 2558 fpde ide la cose maria vitaglscs

bsasescritint6 3egsisisisi y pciade laciottizjorge bautari joseiargasolte 9me1550 14 tue escribenopa perez tapi ctu 1032 de edad ymays 3yogatdel 14121966 l crirto dlanciottig maria luisai solte mayade 1j4 arge s y edadg los dos ul timos hijos de jose ada y argen tina brugnonigliacens 53578 6418 29nizoten adjudicacions e lanciotti autos inscrifton jose suc njuzgeno esi bisecreteno on bidptos b blancail189 511171 del neb107664 gnii del insc prova def e al dorsoill sigue de bucnos aires registro provincia ministerio de economia hacienda de propiedad no la decretoley 11615105 y siguc dorso 4l bp vo bi 1856 fi buml gb sem raoee

2 PaddleOCR

Si cambiamos la librería a paddleOcr, el proceso arroja el siguiente resultado:

MATRICULA
1836 - BAHIA BLANCA (7).-
[...]
(l)LANCIOrTI,José;arg.naturalizado,de -
(q dtH IVLOL ONYO(t)sE SW S& aR VIOd VT aA OONV@
Hipot C 842
origen italiano,may.de edad,jubilado,cas
en lras.c/Ada Bru***ni,M.I xe4.474.119,
vec.de Bahia Blanca.-
pia(7).-2ll 108 del
crit.7/1I/977.Escrib
Comp.Vta.4/III/66.Escrib.Jorge Andrés
19/7/66.-
Anibal R.Lafont (7)-
357123-20/X/66
Palavecino,(7).-99.872,del 18/4/66.-
HIPOT.OBSERY.
79.83l,del
CONSOLIDAD.

28/1II/977
R.y,R.
C.842.
(2)AMPLIACION del Cr6d.Hip.
1
(2)BRU***NI de LAN***TTI,Ada Argentina;arg
2/4b)(1)n°2ll.lO8/966.pox $
VUEL,S,MAT:
46:5.4.
vda.de Jose Lan***ttimmay.de edad,hija de
255.O0O%,a.f/de:BANCOpDE LA
Jose y Maria Vita,L.C.
PCIA.DE BS.AS Escrit.8Y9/66
13695/119/5/200
B vta RI 7407)
Escrib.Nora Pérez Tapia
(7)
/032,may.de edad.-
HSCRIPTO
762
390.947,del 1/12/966.
LAN***TTI,Maria Luisa;arg.,solt.,may,de
1/4
BpV x asor op softu sou4In sop sot pepe
Argentina Brugnoni,L.C.4.552.64l.
Adjudicacion:29/V/70.-En autos"Lanciotti
Jose Sucesion"Juzg.n-3;Secret.n6;Dpto.B
Blanca,-l.l89 del 5/1/71.
INSC.PRGV.-DEF: N.9.1o7.864 del 9/VIII
71
sigue al dorso///

3 Tesseract con el modelo spa

A continuación el resultado de la misma matrícula con la librería Tesseract y el modelo pre entrenado “spa”:

```
= 1836 - BAHIA BLANCA (7).- caras 20: 117 D;MANZ.311-j
;PARC.uno.-
```

[...]

```
(1)LAN***TTI,Josézarg.naturalizado, e (1 )HIPOTECA.
$635.000%,a/£.1e k _ Íp°t_c_g42___

origen itulieno,may.de eJa:i, jubj 0, cas BANCO DE LA PCIA.DE
B3-XS.Es+(2)CANC.TOTAL Hip.b) (7).— c:f¿gc,A_ge°

en lres.c/Ada Bru***ni,!.1 .474.119,
rit.8/6/66.Zecrit.NoraP.Ta-(1), y Ampl.b)(3).Eg- .

vec.de Bahía blance,> 2/ | pia(7).-211.208 del /
terit.7/11/977.Escrib ;

Corp.Vta.4/1II/66Éscrit.Joree Andrés — 19/7/66 inscamTon |
Anibal R.Lafont (7)- 357123-20/L/66

Palavec1 .-99.872,.iel 18/4/66.- .—..,- . QBSERV, d
7<á;8317d$l R7| |Anpl.de Hipots

: CONSOLIDAL 28/11I/97%. J— .842. uN

: * (2)AMPLIACION del Créd.Hip AREN eec

(2)BR/GNONI de LAN***TTI, Ada Argentina;arg 2/4 b)
(1)n9211.108/966.por $ - ' ' y56

vda.de Jose Lencioti izymay.de edud,hija de 255.000% ,a .£/des
BANCO DE LA — —
```

¿ore y Maria Vita,L.C. PCIA.DE BS.A °°nt'¡9/66 3895!1¡—19/54290

%*<;Iorr1,¿or¿e Josejarg. , solt.,N.l.5.506/ 1/4|Escrib.Nora Pérez Tapid (7) —W°v"-7'°°7" .-

32,may .de edad.- 390.947,del 1/12/966 PCRIPTO -

LAN***TTI,Maria Luisajarg.,80l1t.,may,de 1/4 , '__1 "í2

edadi,los dos ultizos h1jos de Jose y Ada

— ;Argentina Prugnoni,L.C.4.552.641. ¿

Adjudicacion:29/V/70.-En autos"Lanciotti

Jose Sucesion"Juzg.n9%3;Secret.no6;Dpto.B £

Blanca,-1.189 del 5/1/71.,- F -

INSC.PROY.—IEF, N.B.107.864 del 9/VIII/ |4

(? '

sigue al dorso/// ¡ I .

4 Tesseract con el modelo eng

= 1836 - BAHIA BLANCA (7).- caras v0: TESD3MANZ.311-j ;PARC.uno.-

[...]

(1)LAN***TTI, José; arg.naturalizado, de (1)HI POTECA. $635.000h,a/fede Cw lepotc.b4ane

origen itulieno,may.de euaii, jub; 6, cas BANCO DE LA PCIA.DE 88-QS.2e+(2)CANC.TOTAL Hip. b) (7) 0+ cen ge

en lres.c/Ada Bru***oni,i.} -474.119, rit.8/6/66.Zecpit.Nora\P.Tat(1),y Ampl.b)(2).E8— °

vec.ue Bahia blance ss e/ | jpia(7).-211.208 del /_ ferit.7/11/977.Escrib

Comp. Vta.4/llL/oGcBscrit.Jorge andrés ". =f 19/7/66 imacareron]| Anibal R.Lafont (7)- 35712 3-20/X/66

Palaveci -- 99.872, iel 18/4/66.- IPO? - QBSERY, CA Beeeyer ayn] |A@pl-de Hipot,

: CONSOLIDAD 28/1II/97%. Js. 842, uh

7 (2)AMPLIACION del Créd.Hip« Ofocee | Caan teen

(2)BRU***NI de LAN***TTI,Ada Argentina;arg 2/4 bd) (1)n°211.108/966.por% - ’ ya es 8 F

vda.de Jose Lanciottigmay.de edad,hija de 255.000%,a-f/des BANCOnDE LA ="e—r.— ee

eore y Maria Vita,L.Cc. PCIA.DE BS.A recri t .BY9/66 3695/1=19/5/200

LATOTTT Jorce Josejarg.,s0lt.,5.1.5.506/ 1/41 Becrib.Nord Pérez Tapif ‘(7) nan RD =

32,may.de edad.- 390.947, del 1/12/966 SFASCRiny 762 |

LAN***TTI,Maria Luisa;arg.,s0lt.,may,de 1/4 , CIN 762

edad,los dos ultizos hijos de Jose y Ada

_ Argentina Brugnoni,L.C.4.552.641.

AGjudicacion:29/V/70.-En autos"Lanciotti

Jose Sucesion"Juzg.n° 3;Secret.n°6;Dpto.B ra

Blanca,-1.189 del 5/I/71.- Fi /

INSC. PRO¥.-DEF, N.B.107.864 del 9/VIII/ }3

Te i

sigue al dorso/// | |

5 Tesseract con el modelo Spa_old

5 1836 - BAHIA BLANCA (7) .- caras to: 115 D;MANZ. 31 1-j ¡PARC.uno.-

[...]

(1) LANCIOTEL ,Josejarg.naturalizado, de (1)HI POTECA. $635. 000%, a/Z<de la? Á ¡arvie Vea

origen itulieno, may.úe euad, jubi 6, cas BANCO DE LA PCILACDE BacAS .Es+(2)cCaNC. TOTAL Hip. b) (7) .- eo

en lras.c/Ada Bru***ni ,.l «474.119, rit.8/6/66.ZscpiviNora \P.Tat(1), y Ampl. b)(2).Es- ©

vec.ue Bahia blanca.> © | lpia(7).+-211.+208 del | lerit.7/11/977.Escrib ;

Comp. Vta. 4/111/6& Éscrib.Jorge Andrés -d 9/7/66 inzgamTon || Anibal R.Lafont (7)- 357l23=20/2/66

Palaveca «99.872, del 18/4/66.- HIPO? ¿QABSER CA | 187222093 7] [Anpl.de Hipots

! C-ONSOLIBAB 28/ 111/97. a — «842. wv>k

- (2)AMPLIACIÓN del Crád «Hips 6 [aria] [age Ecea Ss

(2) BRE***NI de IANCIOTES, Ada Argentinaz are 2/4 b) (l)n°o211.108/966.par $ - ' WE Gs

vda.de Jose Lanciottizmay.de edad,hija de 255. 000%, a «f ¿dez BANCO (DE LA SS

cCose y Waria Vita,L.C. PCIA.JDE BS.<A rócrit.A/9/66 3695/1=19/5/200

MATO TI ¡dores Jdosezarg:,solt.,W.1.5.506/ 1/4 | Escrib.Nara Pérez TapiA (7) tall 2° ©

32, may «de edad .*- 390. 947% del 1/12/9666 Chi T 17162]

LANCIOTTI, Maria Luisajarg:.,solt.,may, de 1/4 ’ <*\1 16)

edad, los dos ultixos hijos de Jose y Ada

Argentina Bru@gnoni,L.Cc4.552.64l. |

Adjudicacion: 29/V/70.-En autos"Lanciotti

Jose Sucesion"duzg.n°3;Secret.no6;Dpto.B 2

Blanca,-1.189 del 5/1/71.- $

INSC. PRES. IEF, NCBClO7.864 del 9/VIII//s

MN’ ¡

si gue al dorso/// ' | |

Referencias Bibliográficas

[1] “Registro de la Propiedad de la Prov. de Bs. As. - DECRETO LEY 11.643”, disponible en https://www.argentina.gob.ar/normativa/provincial/decreto_ley-11643-123456789-0abc-defg-346-1100bvorpyel/actualizacion.

[2] “Registro de la Propiedad de la Provincia de Buenos Aires (RPBA) – Optimización del proceso de Entrada y Salida de Documentos.”, Javier Fabián Bautista. Disponible en: http://sedici.unlp.edu.ar/bitstream/handle/10915/118597/Documento_completo.pdf-PDFA.pdf?sequence=1&isAllowed=y.

[3] Minuta web electrónica. Ref.: DISPOSICION TECNICO REGISTRAL N° 011

[4] ARBA, ref: https://web.arba.gov.ar/

[5] “Decreto ley 9590/1980 de la provincia de Buenos Aires: Incorporación al folio real de todos los partidos de la provincia, estando a cargo del notario interviniente la confección de la matrícula”, disponible en: decreto ley 9590

[6] Walter Sosa Escudero “Big Data”. 5ta edición. Editorial Siglo XXI

[7] Germán Cáseres, Lisandro Delía, Pablo Thomas, Verónica Aguirre, “Generation and Use of a Digest System by Integrating OCR and Smart Searches”, ref: Generation and Use of a Digest System by Integrating OCR and Smart Searches

[8] Eduardo Xamena, Ana Gabriela Maguitman, “Language modeling tools for massive historical OCR post-processing”, ref: Language modeling tools for massive historical OCR post-processing

[9] Federal Agencies Digitization Initiative Still Image Working Group; Rebecca Osborne and Catherine Scott, IBM; Karen Griggs; Erin Rhodes and Steven Puglia, US National Archives and Records Administration. Ref: Technical Guidelines for Digitizing Cultural Heritage Materials: Creation of Raster Image Master Files

[10] “Corrección automática de errores de OCR en documentos semi-estructurados”, Pablo A. Paliza, ref: Corrección automática de errores de OCR en documentos semi-estructurados

[11] Historias de usuarios. Ejemplos y plantillas. https://www.atlassian.com/es/agile/project-management/user-stories#:~:text=usuario%20del%20software.-,Una%20historia%20de%20usuario%20es%20una%20expli

caci%C3%B3n%20general%20e%20informal,un%20valor%20particular%20al%20cliente..
[12] Sepp Hochreiter, Jürgen Schmidhuber, “Long Short-term Memory, 1997, ref: Long Short-term Memory.
[13] A. Graves, M. Liwicki, S. Fernandez, Bertolami, H. Bunke, and J. Schmidhuber, “A Novel Connectionist System for Unconstrained Handwriting Recognition,” IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 31, no. 5, pp. 855–868, May 2008.
[14] S. F. Rashid, F. Shafait, and T. M. Breuel, “Scanning Neural Network for Text Line Recognition,” in DAS, Gold Coast, Australia, Mar. 2012. Disponible en: https://www.researchgate.net/profile/Sheikh-Faisal-Rashid/publication/261273016_Scanning_Neural_Network_for_Text_Line_Recognition/links/54100ffd0cf2df04e75a580e/Scanning-Neural-Network-for-Text-Line-Recognition.pdf
[15] Ray Smith. "An Overview of the Tesseract OCR Engine," in Proceedings of the International Conference on Document Analysis and Recognition, ICDAR, Curitiba, Brazil, IEEE Computer Society, 2007, pp. 629-633. https://static.googleusercontent.com/media/research.google.com/en/us/pubs/archive/33418.pdf
[16] OpenCV OpenCV releases
[17] Manuel Romero, “A packaged and flexible version of the CRAFT text detector and Keras CRNN recognition model”, https://colab.research.google.com/github/mrm8488/shared_colab_notebooks/blob/master/keras_ocr_custom.ipynb#scrollTo=wWdMF9lkKpyI.
[18] Fausto Morales, “Keras-OCR”, https://github.com/faustomorales/keras-ocr.
[19] PaddlePaddle, PaddleOCR, https://github.com/PaddlePaddle/PaddleOCR.
[20] Getting Started with PaddlePaddle: Exploring Object Detection, Segmentation, and Keypoints. Ref: https://learnopencv.com/author/tapas/
[21] Tesseract: an Open-Source Optical Character Recognition Engine. https://www.linuxjournal.com/article/9676
[22] https://github.com/tesseract-ocr/
[23] Mutxu Guiem, “How to train Tesseract 4”, Nov 16, 2019, ref: How to train Tesseract 4
[24] Docker. http://docker.com/.

[25] “Train a new font with Tesseract”, Disponible en Google Colab
[26] jTessBoxEditor, https://vietocr.sourceforge.net/training.html.
[27] Spa_old.traineddata. https://github.com/tesseract-ocr/tessdata/blob/main/spa_old.traineddata.
[28] Scikit-Image 0.22.0 Ref Image Processing for Python
[29] Natural Language Toolkit. https://www.nltk.org/.
[30] DB Engines. https://db-engines.com/en/ranking
[31] Lucene https://lucene.apache.org/
[32] Kibana https://www.elastic.co/es/virtual-events/getting-started-kibana?elektra=en-kibana-page
[33] Elastic Cloud https://www.elastic.co/es/virtual-events/getting-started-kibana?elektra=en-kibana-page
[34] MongoDB MongoDB : todo sobre la base de datos NoSQL orientada a documentos
[35] ¿Qué es MySql?https://www.ionos.es/digitalguide/servidores/know-how/que-es-mysql/#:~:text=MySQL%20es%20un%20sistema%20de,por%20ejemplo%2C%20WordPress%20y%20TYPO3..
[36] MongoDB vs ElasticSearch: the quest of the holy performances. https://blog.quarkslab.com/mongodb-vs-elasticsearch-the-quest-of-the-holy-performances.html
[37] Lorena Talamé, Alejandra Cardoso, Matías Amor, “Comparación de herramientas de procesamiento de textos en español extraídos de una red social para Python”, http://sedici.unlp.edu.ar/bitstream/handle/10915/87854/Documento_completo.pdf-PDFA.pdf?sequence=1&isAllowed=y.
[38] Framework: qué es, para qué sirve y ejemplos. https://unirfp.unir.net/revista/ingenieria-y-tecnologia/framework/#:~:text=%C2%BFQu%C3%A9%20es%20un%20framework?,organizaci%C3%B3n%20y%20desarrollo%20de%20software.
[39] Django Project. Ref: https://www.djangoproject.com/

[40] Flask https://flask.palletsprojects.com/en/3.0.x/.

[41] Flask Extensions. Ref.:http://flask.pocoo.org/extensions/

[42] WSGI Utilities and Reference. Ref.:WSGI Utilities and Reference

[43] Adabas & Natural for Application Modernization. Software AG. https://www.softwareag.com/es_es/platform/adabas-natural.html.

[44] Índice de Titulares del Registro de la Propiedad. https://www.gba.gob.ar/economia/noticias/%C3%ADndice_de_titulares_del_registro_de_la_propiedad.