UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAISO - CHILE

# SISTEMA DE RECONOCIMIENTO DE FOTOGRAFÍAS DE PRODUCTOS

**ERIKO MICHEL KAEMPFER MERCHAN**

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELECTRÓNICO

| | | |
|---|---|---|
| PROFESOR GUÍA | : | SR. WERNER CREIXELL F. |
| PROFESOR CORREFERENTE | : | SR. MAURICIO ARAYA L. |

DICIEMBRE 2022

# AGRADECIMIENTOS

A mi madre que durante toda mi vida ha sido la persona que me ha brindado hasta lo imposible y me ha dado todo lo necesario para desarrollar mi carrera universitaria. Muchas gracias por todo lo que has hecho por mi.

A mi hermana que siempre me ha brindado su apoyo y ha ayudado en todo lo posible.

A mis padrinos por incluirme como parte de su familia y hacerme sentir como un hijo más. Sin tener ninguna obligación me han dado todo y por eso estaré siempre agradecido.

A mis tíos por ayudarme siempre que los he necesitado y brindarme una serie de herramientas que me han ayudado durante todo mi proceso de formación académica.

A mi pareja y mejor amiga por ser un apoyo incondicional y estar ahí siempre para mi, sobre todo en mis peores momentos.

A mis amigos que colaboraron e hicieron del paso por la universidad una gran y divertida instancia. Todas las tardes y noches tanto de estudio como de jugar son momentos que jamás olvidaré

Por último, pero no menos importante, agradezco tanto a mi profesor correferente Mauricio Araya por la buena disposición al solicitarle su participación en este proyecto y a mi profesor guía Werner Creixell que desde el primer momento me dio la motivación que necesitaba para poder sacar adelante este proyecto.

# Índice de Contenidos

**1. Introducción. 1**

**2. Trabajo a desarrollar y resultados esperados 3**
2.1. Objetivos del trabajo. . . . . 3
2.2. Etapas del trabajo . . . . . 3
2.2.1. Algoritmos de reconocimiento óptico de caracteres (OCR). . . . . 3
2.2.2. Estudio y solución de posibles errores. . . . . 4
2.2.3. Módulo de asociación con la base de datos. . . . . 4
2.2.4. Proponer un tipo de salida. . . . . 4
2.3. Evaluaciones a realizar. . . . . 5
2.4. Resultados esperados: . . . . . 5

**3. Estado del Arte 6**
3.1. Visión por computador . . . . . 6
3.2. Reconocimiento óptico de caracteres (OCR) . . . . . 6
3.2.1. Pytesseract . . . . . 7
3.2.2. Métodos para tratar errores típicos de OCR . . . . . 8
3.3. Análisis de trabajos desarrollados por otros autores . . . . . 9
3.3.1. Corrección automática de errores de OCR en documentos semi-estructurados . . . . . 9
3.3.1.1. Resumen del trabajo . . . . . 9
3.3.1.2. Ideas destacadas . . . . . 10

**4. Alternativas posibles de solución del trabajo 11**
4.1. Posibles alternativas . . . . . 11
4.1.1. Alternativa 1: MMOCR . . . . . 11
4.1.1.1. Detección de texto . . . . . 12
4.1.1.2. Reconocimiento de texto . . . . . 13
4.1.1.3. Detección + reconocimiento de texto . . . . . 14
4.1.2. Alternativa 1: Distancia de Levenshtein . . . . . 14
4.1.3. Alternativa 2: Distancia de Levenshtein con peso . . . . . 15
4.1.4. Alternativa 3: Hash de Anagramas (Anagram Hashing) . . . . . 16
4.1.5. Alternativa 4: Clave de similitud (Similarity key) . . . . . 16
4.2. Ventajas y desventajas de las alternativas . . . . . 17
4.2.1. Alternativa 1: MMOCR . . . . . 17
4.2.2. Alternativa 1: Distancia de Levenshtein . . . . . 17
4.2.3. Alternativa 2: Distancia de Levenshtein con peso . . . . . 18
4.2.4. Alternativa 3: Hash de Anagramas (Anagram Hashing) . . . . . 18
4.2.5. Alternativa 4: Clave de similitud (Similarity key) . . . . . 19
4.3. Características de comparación. . . . . 19

**5. Propuesta de solución 20**

5.1. Importancia de las características de comparación para las soluciones . . . . . 20
5.2. Mejor alternativa según su conjunto de características . . . . . 20

**6. Desarrollo de la solución. 22**
6.1. Desarrollo de las etapas planteadas . . . . . 22
6.1.1. Elegir el modelo de reconocimiento que se utilizará . . . . . 22
6.1.2. Crear clases de equivalencia . . . . . 25
6.1.3. Implementar lo métodos de corrección de errores del OCR . . . . . 26
6.1.3.1. Distancias de Levenshtein 1 y 2 . . . . . 26
6.1.3.2. Considerar la distancia de Levenshtein menor . . . . . 28
6.1.3.3. Considerar la distancia de Levenshtein menor y utilizar claves de similitud . . . . . 29

**7. Conclusiones y posibles trabajos a futuro 30**
7.1. Conclusiones . . . . . 30
7.2. Posibles trabajos a futuro . . . . . 31

**Bibliografía 32**

**A. Anexo 34**
A.1. Módulo de asociación con la base de datos . . . . . 34
A.2. Función para obtener todas las cadenas de caracteres a una distancia de Levenshtein de 1 . . . . . 34
A.3. Función para obtener todas las cadenas de caracteres a una distancia de Levenshtein de 2 . . . . . 35
A.4. Función para obtener la distancia de Levenshtein entre dos cadenas de caracteres . . . . . 35

# Índice de Tablas

4.1. Modelos de detección . . . . . 12
4.2. Comparación entre los modelos de reconocimiento. . . . . 13
4.3. Matriz de distancia de Levenshtein entre dos palabras. . . . . 15
4.4. Características comparación . . . . . 19

5.1. Porcentajes de la mejor solución . . . . . 21

6.1. Comparación entre los modelos de reconocimiento. . . . . 22
6.2. Comparación entre los modelos de reconocimiento en datos de productos reales. . . . . 24
6.3. Comparación entre el mismo modelo pero variando la diferencia. . . . . 27
6.4. Comparación utilizando la distancia Levenshtein más próxima pero variando la diferencia. . . . . 28
6.5. Comparación utilizando la distancia Levenshtein más próxima pero variando la diferencia. . . . . 29

7.1. Comparación de tiempos de ejecución entre los distintos modelos . . . . . 30
7.2. Comparación de tiempos de ejecución entre los distintos métodos . . . . . 30

# Índice de Figuras

1.1. Diagrama del sistema a desarrollar. . . . . 2

3.1. Flujo de algoritmo Pytesseract. . . . . 8

4.1. Ejemplo de detección de textos . . . . . 12
4.2. Ejemplo de reconocimiento de textos . . . . . 13
4.3. Ejemplo de reconocimiento + detección de textos . . . . . 14

6.1. Gráfica de la comparación entre modelos de reconocimiento aplicado a imágenes reales. . . . . 24
6.2. Arquitectura del modelo ABINet (1). . . . . 25
6.3. Comparación entre el método de Dist de Lev 1 y Dist de Lev 2 . . . . . 26
6.4. Cantidad promedio de posibles productos detectados . . . . . 27
6.5. Cantidad promedio de posibles productos detectados para el método de distancia de Levenshtein más próxima . . . . . 28
6.6. Cantidad promedio de posibles productos detectados para el método de distancia de Levenshtein más próxima con clave de similitud . . . . . 29

# RESUMEN

En el siguiente proyecto de título se presentará un sistema que busca reconocer distintos productos de la industria del retail mediante imágenes, pero sin la necesidad de tener que entrenar un modelo de reconocimiento especifico de estos productos. Para lograr esto se utilizará un modelo de reconocimiento óptico de caracteres (OCR) para así detectar las palabras que se encuentren en las imágenes de productos y mediante una base de datos que contiene las palabras de cada uno de los productos se realizará una asociación. A simple vista, sería fácil asociar palabras entre si, pero uno de los grandes problemas que poseen los modelos OCR es que sus predicciones pueden contener una serie de errores. Es por esto que se buscará implementar un método que mejore las predicciones hechas por el modelo.

# ABSTRACT

In the following final degree project, a system that seeks to recognize different products in the retail industry through images will be presented, but without the need to train a specific recognition model for these products. To achieve this, an optical character recognition (OCR) model will be used to detect the words found in the product images and an association will be made through a database that contains the words of each of the products. At first glance, it would be easy to associate words with each other, but one of the big problems with OCR models is that their predictions can contain a series of errors. This is why we will seek to implement a method that improves the predictions made by the model.

# 1 | Introducción.

El auge de la inteligencia artificial ha provocado que hoy en día sea cada vez más común la automatización de tareas que hasta hace un tiempo eran trabajos hechos por personas. Un ejemplo de lo antes mencionado es el control de stock, lo cual implica una serie de problemas para las empresas. Los problemas vienen a partir de las grandes cantidades de productos que tienen las empresas, lo que implica grandes cantidades de empleados para realizar los recuentos, lo que a su vez conlleva a grandes cantidades de horas y sueldos por pagar. Y sin contar lo propenso a errores que son los trabajos hechos por personas. Es por esto que las grandes organizaciones cada vez se interesan más en requerir los servicios de otras empresas que se dedican a trabajar en la automatización de tareas, como es el caso de Soluciones IA Spa.

La empresa para la cual se hará el sistema trabaja con herramientas de IA (inteligencia artificial). En particular, se especializa en aplicaciones que permiten identificar productos en base a diversos análisis de imágenes. A partir de esto es que surge un nuevo proyecto, donde un punto fundamental es el reconocer caracteres y palabras en fotografías de diversos productos comercializados en la industria del retail.

En este proyecto de título se abordará una solución al problema antes mencionado, donde se implementará un sistema que sea capaz de reconocer textos y caracteres, específicamente que pueda leer etiquetas de productos y así detectar las palabras más importantes de cada uno. Luego se elaborará un módulo de asociación con la base de datos lo más preciso posible, de modo que se logré un equilibrio entre tiempo de ejecución y porcentaje de aciertos. Para realizar la asociación de productos la empresa cuenta con una base de datos, donde están escritas las palabras más relevantes y están asociadas al SKU (Stock Keeping Unit) de cada producto. A continuación se propone un diagrama de como funcionará el sistema:

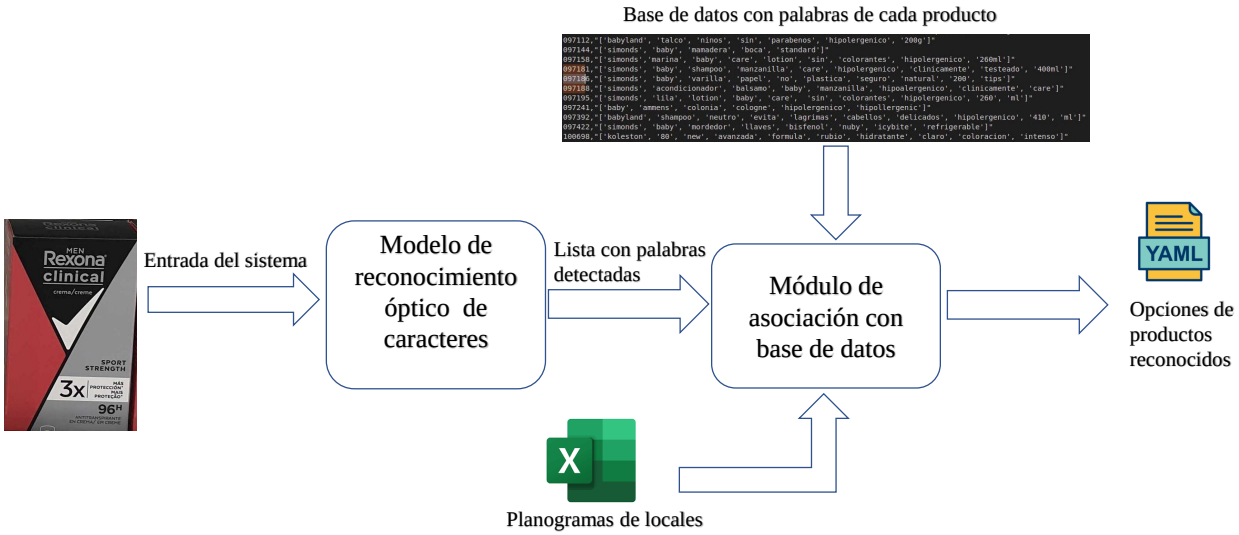



**Figura 1.1:** Diagrama del sistema a desarrollar.

En el diagrama se puede observar que la entrada al sistema son recortes de tal manera que queda un producto por imagen. Estos pasan por el modelo de reconocimiento óptico de caracteres, el cual entrega una lista con las palabras detectadas. Esta lista tendrá palabras que no están detectadas completamente bien, debido a que este tipo de modelos es propenso a tener errores relacionados a la imagen tomada o al lugar donde se toma (fotos borrosas, problemas de luminosidad, tamaños de las letras, etc.). La lista ingresará al módulo de asociación junto a la base de datos y los plano gramas de los locales. Estos planogramas contienen los diferentes productos separados por categorías, como pueden ser bebidas y aguas, cremas corporales, desodorantes corporales, etc. Esto servirá al momento de hacer las relaciones con los productos, ya que al separarlo por categorías se tendrá que evaluar una cantidad menor de veces. Por ultimo, se tendrá un archivo YAML que corresponde a la salida del sistema. En este archivo se encontrarán todas las posibles opciones de productos reconocidos.

# 2 Trabajo a desarrollar y resultados esperados

## 2.1. Objetivos del trabajo.

El objetivo principal del trabajo de título es reconocer productos a través del texto que poseen en sus etiquetas.

Para lograr el objetivo principal se deben cumplir primero los siguientes objetivos, los cuales son requisitos de la empresa:

- Implementar un modelo de reconocimiento de caracteres que no tarde más de 5 segundos por cada imagen.
- Realizar un módulo de asociación, donde se hará coincidir la lista de palabras detectadas con las palabras que se encuentran en la base de datos.
- Alcanzar al menos un 75 % de aciertos en los emparejamientos.
- No sobrepasar cinco productos entre las posibilidades devueltas por el sistema.

## 2.2. Etapas del trabajo

El trabajo por desarrollar a lo largo del proyecto de titulación puede descomponerse en los siguientes pasos:

### 2.2.1. Algoritmos de reconocimiento óptico de caracteres (OCR).

Para lograr el reconocimiento de productos, a través de imágenes, lo más intuitivo podría ser generar una base de datos en donde se tengan etiquetadas imágenes de los productos y utilizar un clasificador multiclase para reconocerlos. El problema que tiene este método es que generar estas bases de datos no es trabajo sencillo, mas bien, todo lo contrario. La cantidad de productos que se manejan en las industrias del retail son del orden de decenas de miles, por lo que generar una base de datos con la suficiente cantidad de imágenes para que un modelo de reconocimiento pueda detectar las diferencias entre productos es una tarea bastante costosa y extensa si es que se empieza desde cero.

Investigando con lo que cuenta la empresa para la que se realizará este proyecto es que se posee una base de datos en la cual se tiene las palabras más importantes, en orden jerárquico, de cada uno de los productos a analizar. A partir de esto es que se buscan posibles alternativas, en donde el objetivo es optimizar los recursos que se poseen. Después de explorar soluciones que no involucren crear nuevas bases de datos desde cero es que surge la idea de utilizar un modelo de reconocimiento óptico de caracteres (OCR por sus siglas en inglés).

Para esto se utilizarán y probarán una serie de modelos que se especifican en (2). Estos modelos se pondrán a prueba para ver cual cumple de mejor manera los objetivos propuestos y a partir de esto escoger el indicado.

### 2.2.2. Estudio y solución de posibles errores.

El nivel de efectividad de los algoritmos OCR muchas veces se ve afectado por factores extrínsecos relacionados a la imagen tomada y al lugar donde se toma. A continuación se mencionan algunos de los que ocurren comúnmente:

- Letra borrosa o poco nítida.
- Manchas en las etiquetas.
- Transparencia en las etiquetas.
- Palabras fragmentadas.
- Palabras solapadas.
- Tipografías extrañas o fuera de uso.
- Dimensiones de las palabras.
- Baja resolución de la imagen.
- Problemas de luminosidad.

Es por esto que hay que estudiar como afectan estos errores y proponer métodos para que esto no conlleve a una baja precisión.

### 2.2.3. Módulo de asociación con la base de datos.

Después de proponer todos los métodos estudiados para lograr un correcto funcionamiento del sistema estos se llevarán a cabo en este módulo, con el objetivo de lograr corregir la mayor cantidad errores posibles. Además, se realizará un sistema de puntajes con el cual se buscará hacer el emparejamiento entre las palabras que se detectaron de un producto y las palabras de cada producto en la base de datos.

### 2.2.4. Proponer un tipo de salida.

Por último, pero no menos importante, hay que proponer un tipo de salida. Los productos que van a ser analizados por el sistema pueden llegar a ser prácticamente iguales ante la visión de un modelo. Algunas veces la única diferencia entre un producto y otro es el sabor o aroma, lo cual si es que en algún caso no se logra detectar esa única palabra de diferencia el sistema se confundiría

entre los dos. Es por esto que la idea de este sistema no es simplemente reconocer el producto que se está viendo en la imagen, si no devolver como salida los posibles productos con mayor nivel de emparejamiento entre las palabras detectadas y entregar esta información a un siguiente módulo que busqué una solución a este problema.

## 2.3. Evaluaciones a realizar.

Para realizar todas las pruebas necesarias la empresa cuenta con varias fotos, las cuales equivalen a fotos reales que esperan ser tomadas por la aplicación funcional. A partir de estas se harán pruebas con datasets de imágenes, separados por categorías para así evaluar posibles problemas en las bases de datos. Al no ser un modelo entrenado directamente con las imágenes cada prueba que se haga serán siempre datos nuevos para el modelo. Cada imagen deberá pasar por el sistema y a partir de eso se evaluará la efectividad de este, tanto en tiempo de ejecución como en si encontró dentro de sus posibilidades el producto real.

## 2.4. Resultados esperados:

El resultado óptimo sería poder realizar un algoritmo de reconocimiento de productos que logre obtener un 100 % de efectividad independiente del tipo de iluminación, posición del producto y sin la necesidad de un siguiente módulo o supervisión humana para que pueda efectuar esto.

Debido a que las fotos no siempre son las óptimas y considerando el factor humano al momento de tomarlas, se da por entendido que es imposible lograr el resultado ideal. Otros factores a tener en cuenta es que estas se toman en góndolas donde pueden haber flejes que tapen palabras importantes de los productos, por lo que no siempre se ven todas las palabras.

Finalmente se espera encontrar un sistema de reconocimiento óptico de caracteres, el cual equilibre lo mejor posible tanto tiempo de ejecución, como la precisión al momento de acertar su predicción. Después de esto se espera realizar un módulo de emparejamiento, con el objetivo de que pueda emparejar el máximo número de palabras posibles, por más que estas puedan haber sido reconocidas de manera errónea.

# 3 | Estado del Arte

El objetivo de este trabajo de título es desarrollar un sistema automático para reconocer productos mediante imágenes. Por lo tanto, usaremos tecnologías del área de visión por computador, inteligencia artificial, OCR, métodos para tratar los posibles errores de los modelos OCR, entre otros. A continuación, se presentan los temas principales a investigar y desarrollar en este trabajo de título.

## 3.1. Visión por computador

El término visión por computador, define el área genérica de investigación donde se interpretan las características del mundo 3D real en datos métricos mediante el procesamiento de imágenes 2D. Las aplicaciones básicas de la visión por computador incluye construcción y vigilancia de modelos 3D, interacción con el entorno, análisis de imágenes, etc. La realización de estas aplicaciones requiere la ejecución de varios algoritmos, que procesan imágenes 2D y proporcionan información 3D. Algunos de estos algoritmos realizan reconocimiento de objetos, seguimiento de objetos, estimación de pose, estimación de movimiento, flujo óptico y reconstrucción de escenas (3).

## 3.2. Reconocimiento óptico de caracteres (OCR)

El reconocimiento óptico de caracteres u OCR es la traducción electrónica de textos manuscritos, mecanografiados o texto impreso en imágenes traducidas automáticamente. Es ampliamente utilizado para reconocer y buscar texto en documentos electrónicos o para publicar el texto en un sitio web. (4).

Todos los algoritmos de OCR tienen la finalidad de poder diferenciar un texto de una imagen cualquiera. Para lograr esto llevan a cabo la siguiente secuencia:

- Binarización o caracterización: Es una técnica que consiste en la realización de un barrido en la matriz de la imagen digital, por medio de bucles o recursividad, con el fin de que el proceso produzca la reducción de la escala de grises a dos únicos valores. Negro (= 0) y blanco (= 255), o lo que es lo mismo, un sistema binario de ausencia y presencia de color 0-1. La comparación de cada píxel de la imagen viene determinada por el umbral de sensibilidad (valor T = Threshold). Por ejemplo, los valores que sean mayores que el umbral toman un valor 255 (blanco) y los menores 0 (negro) (5).
- Fragmentación o segmentación de la imagen: Este es el proceso más costoso y necesario para el posterior reconocimiento de caracteres. La segmentación de una imagen implica la detección

mediante procedimientos de "etiquetado determinista" o estocástico de los contornos o regiones de la imagen, basándose en la información de intensidad o información espacial.

Permite la descomposición de un texto en diferentes entidades lógicas, que han de ser suficientemente invariables, para ser independientes del escritor, y suficientemente significativas para su reconocimiento.

No existe un método genérico para llevar a cabo esta segmentación de la imagen que sea lo suficientemente eficaz para el análisis de un texto. Aunque las técnicas más utilizadas son variaciones de los métodos basados en proyecciones lineales.

Una de las técnicas más clásicas y simples para imágenes de niveles de grises consiste en la determinación de los modos o agrupamientos (clústeres) a partir del histograma, de tal forma que permitan una clasificación o umbralización de los píxeles en regiones homogéneas.

- Adelgazamiento de los componentes: Una vez aislados los componentes conexos de la imagen, se les tendrá que aplicar un proceso de adelgazamiento para cada uno de ellos. Este procedimiento consiste en ir borrando sucesivamente los puntos de los contornos de cada componente de forma que se conserve su tipología.

  La eliminación de los puntos ha de seguir un esquema de barridos sucesivos para que la imagen continúe teniendo las mismas proporciones que la original y así conseguir que no quede deforme.

- Comparación con patrones: En esta etapa, se comparan los caracteres obtenidos anteriormente con unos teóricos (patrones) almacenados en una base de datos. El buen funcionamiento del OCR se basa en gran medida en una buena definición de esta etapa.

### 3.2.1. Pytesseract

Pytesseract o Python-tesseract es una herramienta de reconocimiento óptico de caracteres (OCR) para Python. Esta herramienta lee y reconoce el texto en imágenes y lo transforma en lenguaje de máquinas. Python-tesseract es en realidad una clase contenedora o un paquete para el motor Tesseract-OCR de Google, el cual es una de las herramientas de OCR más conocidas. También es útil y se considera como un script de invocación independiente para tesseract, ya que puede leer fácilmente todos los tipos de imágenes compatibles con las bibliotecas de imágenes Pillow y Leptonica, que incluyen principalmente:

- jpg
- png
- gif
- bmp
- tiff, etc.

A continuación, se presenta un diagrama de flujo representativo de la herramienta Pytesseract en la Figura 3.1.

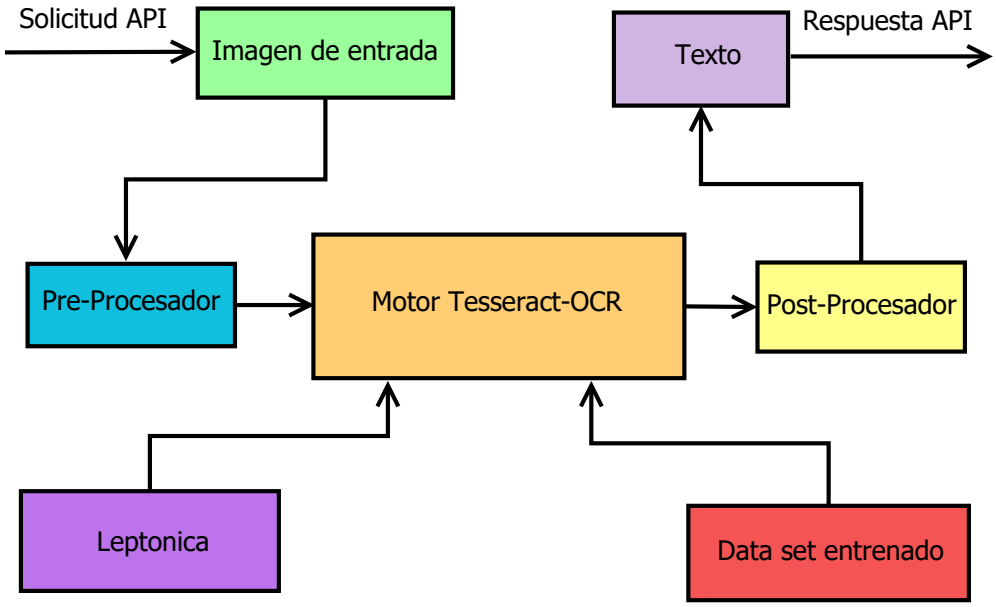



**Figura 3.1:** Flujo de algoritmo Pytesseract.

### 3.2.2. Métodos para tratar errores típicos de OCR

El nivel de efectividad de los algoritmos OCR muchas veces se ve afectado por factores extrínsecos relacionados a la imagen tomada y al lugar donde se toma. A continuación se mencionan algunos de los que ocurren comúnmente:

- Letra borrosa o poco nítida.
- Transparencia en las etiquetas.
- Palabras fragmentadas.
- Palabras solapadas.
- Tipografías extrañas o fuera de uso.
- Dimensiones de las palabras.
- Baja resolución de la imagen.
- Problemas de luminosidad.

Es por esto que se presentan los siguientes métodos para mejorar el reconocimiento de palabras (6).

- Distancia de Levenshtein
- Distancia de Levenshtein con peso
- Hash de Anagramas (Anagram Hashing)
- Clave de similitud (Similarity key)

Se entrará más en profundidad en estos métodos desde la sección 4.1.2 a la sección 4.1.5

## 3.3. Análisis de trabajos desarrollados por otros autores

### 3.3.1. Corrección automática de errores de OCR en documentos semi-estructurados

#### 3.3.1.1. Resumen del trabajo

En este trabajo se presenta la tarea realizada para corregir automáticamente texto generado por un OCR desde un archivo digital realizado para preservar documentos muy antiguos que datan entre los años 1976 y 1983. Según el autor estos documentos son únicos tanto en su contenido, estructura como en su estado de conservación. El proceso a través de un algoritmo de OCR para el tipo de documentos con los que se trabaja en este informe da paso a que se den equivocaciones en la detección de algunos caracteres. Estos errores en su mayoría vienen dado por el estado de los documentos. Algunos ejemplos de errores que suelen cometer estos algoritmos es al momento de diferencia letras como la *e* y la *c* o también con la *u* y la *v*, donde dependiendo de la tipografía de la letra estas pueden verse prácticamente igual ante la visión de los computadores. Es por esto que en el trabajo se plantean métodos para trabajar los errores típicos del OCR.

Los métodos planteados en el trabajo desarrollado son cuatro y son: Distancia de Levenshtein, Distancia de Levenshtein con peso, Hash de Anagramas (Anagram Hashing) y Clave de similitud (Similarity key). El primera corresponde a un método para obtener el numero de operaciones necesarias para transformar una cadena de caracteres en otra. El segundo corresponde a una variación del primero, pero con la diferencia de que se puede asignar un cierto peso dependiendo del tipo de error cometido. Por ejemplo el intercambiar las letras *e* con la *c* tendrían un peso menor que el intercambiar la letra *e* por la *k*, debido a que el error cometido en el primer ejemplo es más común ya que esas letras son muy similares para un procesador OCR, debido a su forma. El tercer método sirve para producir un número lo suficientemente grande para poder identificar palabras que tengan los mismos caracteres en común. Por último, se tiene el método de clave de similitud donde lo que se busca es mapear palabras a claves con el objetivo de que las palabras que son similares tengan una clave idéntica o por lo menos similar. Este último método depende mucho del procesador OCR utilizado y del corpus de datos con el que se trabaja, ya que de esto depende la manera de formar estas clases de equivalencia.

Para el desarrollo de este trabajo se utilizó la estrategia de armar clases de equivalencias (aplicando el método de clave de similitud) donde se armaron las siguientes clases: [fiíjklrtIJLT1!] [abdgoopqOQ690] [éecCG] [vxyVYX] [sS5] [zZ] [EFPRK] [úu], lo cual significa que un carácter de una clase solo puede ser reemplazado por cualquiera de los otros caracteres de la misma clase. A su vez se arma un diccionario donde se almacenan las palabras posibles en estos documentos y su frecuencia. La función de este diccionario es comprobar tanto las palabras validas como ordenar por mayor frecuencia la lista de candidatos para corregir la palabra.

#### 3.3.1.2. Ideas destacadas

A partir del estudio de este trabajo lo más destacable al momento de realizar el tema de esta memoria son los métodos que se proponen para la corrección de palabras. Específicamente una mezcla entre clave de similitud y la distancia de Levenshtein parece una buena primera prueba al momento de corregir estas palabras.

# 4 | Alternativas posibles de solución del trabajo

## 4.1. Posibles alternativas

Las alternativas de solución se dividirán en dos partes: alternativas para el modelo OCR a utilizar y alternativas de métodos para mejorar la salida del OCR. Las primeras alternativas que veremos serán las del sistema OCR.

### 4.1.1. Alternativa 1: MMOCR

La primera y única alternativa que se presentará de algoritmo de reconocimiento de texto es la de MMOCR (2). Esta se escogió debido a que la empresa para la que se realiza este proyecto la propuso como la alternativa que más les atraía. A pesar de esto se presenta igualmente un análisis de esta.

MMOCR es una herramienta open-source que está basada en Pytorch y mmdetection para la detección de texto, el reconocimiento de texto y las tareas posteriores correspondientes incluida la extracción de información clave. Este es parte del proyecto OpenMMLab.

Para utilizar esta herramienta se necesita cumplir una serie de prerrequisitos, los cuales son:

- tener un sistema operativo (SO) Linux | Windows | macOS
- Python 3.7
- PyTorch 1.6 o mayor
- torchvision 0.7.0
- CUDA 10.1
- NCCL 2
- GCC 5.4.0 o mayor
- MMCV
- MMDetection

Después de esto se debe seguir una serie de pasos para la instalación (2)

Esta herramienta proporciona una serie de opciones para ser utilizada, entre estas se encuentran las siguientes

#### 4.1.1.1. Detección de texto

Como su nombre lo indica, la detección de textos se basa en encontrar las posiciones en donde se haya cualquier tipo de texto, sin necesariamente reconocer que dice el texto.

Para realizar la detección de texto se tienen una serie de modelos ya entrenados sobre dos sets, los cuales son ICDAR2015 y CTW1500. Estos sets contienen distintas imágenes con texto en escenas. Los resultados de cada uno de los modelos se aprecian en la Tabla 4.1.

| Metodo | Training Set | Training Set | Epochs | Test Size | Recall | Precision | Hmean |
|---|---|---|---|---|---|---|---|
| DBNet_r18 (7) | ICDAR2015 Train | ICDAR2015 Test | 1200 | 736 | 0.731 | 0.871 | 0.795 |
| DBNet_r50dcn (7) | ICDAR2015 Train | ICDAR2015 Test | 1200 | 1024 | 0.814 | 0.868 | 0.840 |
| DBNetpp_r50dcn (8) | ICDAR2015 Train | ICDAR2015 Test | 1200 | 1024 | 0.822 | 0.901 | 0.860 |
| DRRG (9) | CTW1500 Train | CTW1500 Test | 1200 | 640 | 0.822 | 0.858 | 0.840 |
| FCENet (10) | CTW1500 Train | CTW1500 Test | 1500 | 736 | 0.828 | 0.875 | 0.851 |
| MaskRCNN (11) | ICDAR2015 Train | ICDAR2015 Test | 160 | 1920 | 0.783 | 0.872 | 0.825 |
| PANet (12) | CTW1500 Train | CTW1500 Test | 600 | 640 | 0.776 | 0.838 | 0.806 |
| PSENet-4s (13) | CTW1500 Train | CTW1500 Test | 600 | 1280 | 0.728 | 0.849 | 0.784 |
| TextSnake (14) | CTW1500 Train | CTW1500 Test | 1200 | 736 | 0.795 | 0.840 | 0.817 |

**Tabla 4.1:** Modelos de detección

A continuación, se muestra un ejemplo de como es una detección de texto

**Figura 4.1:** Ejemplo de detección de textos

Como se aprecia en la figura 4.1 se detecta el texto, dejándolo enmarcado en una zona verde. las coordenadas de esta zona también pueden ser recibidas como una salida de la función.

#### 4.1.1.2. Reconocimiento de texto

Como su nombre lo indica, el reconocimiento de textos se basa en transformar el texto que aparece en una imagen en lenguaje de máquinas, sin necesariamente detectar donde está el texto.

Para realizar el reconocimiento de texto se tienen una serie de modelos ya entrenados sobre varios tipos de sets, los cuales se separan en sets con texto regular y sets con texto irregular. El texto regular corresponde al texto que cuenta con secuencias de caracteres que están en linea recta. El texto irregular, en cambio, es más desafiante debido a que posee formas curvas y distorsiones de perspectiva (15). Estos modelos son los presentados en la Tabla 4.2

| Método | Texto regular | Texto regular | Texto regular | Texto irregular | Texto irregular | Texto irregular |
|---|---|---|---|---|---|---|
| - | IIIT5K | SVT | IC13 | IC15 | SVTP | CT80 |
| ABINet (1) | 95.7 % | 94.6 % | 95.7 % | 85.1 % | 90.4 % | 90.3 % |
| CRNN (16) | 80.5 % | 81.5 % | 86.5 % | 54.1 % | 59.1 % | 55.6 % |
| MASTER (17) | 95.27 % | 89.8 % | 95.17 % | 77.03 % | 82.95 % | 89.93 % |
| NRTR (18) | 95.2 % | 90.0 % | 94.0 % | 74.1 % | 79.4 % | 88.2 % |
| RobustScanner (19) | 95.1 % | 89.2 % | 93.1 % | 77.8 % | 80.3 % | 90.3 % |
| SAR (20) | 95.2 % | 88.7 % | 92.4 % | 78.2 % | 81.9 % | 89.6 % |
| Satrn (21) | 96.1 % | 93.5 % | 95.7 % | 84.1 % | 88.5 % | 90.3 % |
| CRNN-STN (22) | 80.8 % | 81.3 % | 85.0 % | 59.6 % | 68.1 % | 53.8 % |
| SEG (23) | 90.9 % | 81.8 % | 90.7 % | - | - | 80.9 % |

**Tabla 4.2:** Comparación entre los modelos de reconocimiento.

A continuación, se muestra un ejemplo de como es un reconocimiento de texto

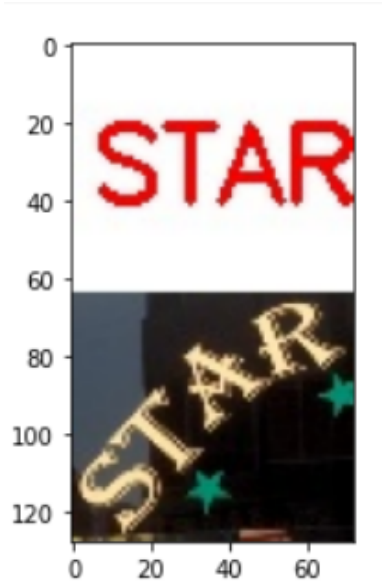



**Figura 4.2:** Ejemplo de reconocimiento de textos

Como se aprecia en la figura 4.2 se reconoce el texto, mostrándolo en una imagen aparte y en color rojo, mas no se conoce la posición en la que se encuentra.

#### 4.1.1.3. Detección + reconocimiento de texto

Como su nombre lo indica, este método consiste en hacer una mezcla entre los dos puntos mencionados anteriormente. Al unir los dos métodos anteriores la robustez de este se ve elevada considerablemente, por lo que actualmente se ve como la mejor opción para abordar el problema. A continuación, se muestra un ejemplo de como se ve este método implementado.

**Figura 4.3:** Ejemplo de reconocimiento + detección de textos

Aquí se puede observar que se reconoce el texto y además se detecta la posición en la que se encuentra.

Ahora se presentarán las alternativas de métodos para tratar la salida del OCR

### 4.1.2. Alternativa 1: Distancia de Levenshtein

Una forma de medir la similitud de dos cadenas de caracteres es mediante la “Distancia de Levenshtein”. Esta se define como el número mínimo de operaciones requeridas para transformar una cadena de caracteres en otra. Se entiende por operación una inserción, eliminación o la sustitución de un carácter. Por ejemplo, la distancia de Levenshtein entre “casa” y “carpa” es de 2 porque se necesitan al menos dos ediciones elementales para cambiar uno en el otro y entre casa y cara es de 1.

- casa → cara (sustitución de ’s’ por ’r’)
- cara → carpa (inserción de ’p’ entre ’r’ y ’a’)

Para calcular la distancia de Levenshtein de dos cadenas $x := x_1...x_n$ y $y := y_1...y_m$ de manera eficiente, se puede utilizar el algoritmo de Wagner-Fischer (24), que utiliza programación dinámica. El algoritmo utiliza una matriz de distancia $D : (n + 1) \times (m + 1)$. Sea - el símbolo neutro (no carácter), entonces cada entrada de la matriz $D_{i,j}$ se puede calcular recursivamente de la siguiente manera:

$$
\begin{aligned}
D_{0,0} &= 0 \\
D_{i,0} &= D_{i-1,0} + cost(x_i, -), 1 \leq i \leq n \\
D_{0,j} &= D_{0,j} + cost(-, y_i), 1 \leq j \leq m \\
D_{i,j} &= min \begin{Bmatrix} d_{i-1,j} + cost(x_i, -) \\ d_{i,j-1} + cost(-, y_i) \\ d_{i-1,j-1} + cost(x_i, y_i) \end{Bmatrix}
\end{aligned} \tag{4.1}
$$

Para la distancia de Levenshtein clásica se utiliza la siguiente función de costo:

$$
cost(x_i, y_j) = min \begin{Bmatrix} 0 & si & x_i = y_i \\ 1 & caso & contrario \end{Bmatrix} \tag{4.2}
$$

Ahora se presenta un ejemplo de las distancias entre las palabras calculadora y computadora:

| x/y | | C | A | L | C | U | L | A | D | O | R | A |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| C | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| O | 2 | 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 7 | 8 | 9 |
| M | 3 | 2 | 2 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 8 | 9 |
| P | 4 | 3 | 3 | 3 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 9 |
| U | 5 | 4 | 4 | 4 | 4 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| T | 6 | 5 | 5 | 5 | 5 | 4 | 4 | 5 | 6 | 7 | 8 | 9 |
| A | 7 | 6 | 5 | 6 | 6 | 5 | 5 | 4 | 5 | 6 | 7 | 8 |
| D | 8 | 7 | 6 | 6 | 7 | 6 | 6 | 5 | 4 | 5 | 6 | 7 |
| O | 9 | 8 | 7 | 7 | 7 | 7 | 7 | 6 | 5 | 4 | 5 | 6 |
| R | 10 | 9 | 8 | 8 | 8 | 8 | 8 | 7 | 6 | 5 | 4 | 5 |
| A | 11 | 10 | 9 | 9 | 9 | 9 | 9 | 8 | 7 | 6 | 5 | 4 |

**Tabla 4.3:** Matriz de distancia de Levenshtein entre dos palabras.

### 4.1.3. Alternativa 2: Distancia de Levenshtein con peso

El método antes presentado, la "Distancia de Levenshtein", puede ser modificado para actuar frente a diferentes tipo de errores. Cada sustitución de caracteres, como también las inserciones y eliminaciones pueden tener diferentes valores. Con respecto a los errores de OCR, se pueden utilizar funciones de costo complejas, por ejemplo, la sustitución del carácter 'e' por 'c' podría tener menor costo, ya que son muy similares para un procesador de OCR dado que tienen una forma similar. Otro ejemplo de esto es el carácter 'v' con el carácter 'y' si es que la etiqueta o envase se ve brillante o la imagen no tiene buena calidad y la parte inferior de la 'y' esta borrosa. Asignándole un menor

costo a caracteres similares se puede hacer que por ejemplo la distancia entre la palabra Lev y Ley sea menor que la distancia entre Lev y Leo.

### 4.1.4. Alternativa 3: Hash de Anagramas (Anagram Hashing)

El método Hash de Anagramas usa principalmente una tabla de hash y una función de hash; esta función está diseñada para producir un número lo suficientemente grande para poder identificar palabras que tengan los mismos caracteres en común. Tales palabras son llamadas anagramas. En términos generales un anagrama es una palabra que está formada desde otra palabra solamente intercambiando caracteres. Por ejemplo, anagramas de la palabra ROMA son: ARMO, RAMO, AMOR, MORA, y todas las palabras que se puedan formar combinando estas cuatro letras.

Formalmente la función hash se define de la siguiente manera:

$$hash(p) := \sum_{i=1}^{|p|} int(c_i)^n, \tag{4.3}$$

donde p es la palabra, cada c corresponde a un caracter de la palabra, la función int() mapea cada caracter a un valor numérico y n corresponde a una potencia que se eleva cada valor.

La idea de esta función es que arroja el mismo valor para las palabras que tienen los mismos caracteres.

### 4.1.5. Alternativa 4: Clave de similitud (Similarity key)

Esta técnica consiste en mapear palabras a claves, de manera que las palabras que son similares tengan una clave idéntica o por lo menos similar. La idea de este calculo es que caracteres que probablemente pueden ser confundidos durante el proceso de OCR, caen en la misma clase de equivalencia. Las clases de equivalencia buscan adaptarse tanto al procesador OCR como a los tipos de fuentes que utilizan los caracteres a analizar. Es por esto que al crear las clases de equivalencia es fundamental tener claro el procesador que se utilizará y el corpus de datos a analizar. Cada una de estas clases debería tener los caracteres que son más probables de confundir entre si, lo que normalmente se decide por la forma que estos tienen. Por ejemplo, los caracteres e y c deberían estar en la misma clase de equivalencia. Otro caso es el de la u y la v, los cuales también deberían estar en la misma clase de equivalencia, pero en una distinta al ejemplo anterior.

## 4.2. Ventajas y desventajas de las alternativas

Primero se presentarán los conceptos que se utilizarán para medir las ventajas y desventajas de cada uno de los métodos. Estos son los siguientes.

- Tiempo: Hace referencia al tiempo que se tardará en implementar el método en un código.
- Generalización: Hace referencia a la capacidad que tiene el método al momento de desenvolverse con palabras nunca antes vistas.
- Efectividad: Hace referencia a que tanto porcentaje de efectividad tiene el método frente a una cadena de caracteres.
- Robustez: Hace referencia a la capacidad del método frente a situaciones que podrían generar un mayor conflicto.

A partir de estos conceptos, primero se analizarán las ventajas y desventajas del modelo OCR a utilizar.

### 4.2.1. Alternativa 1: MMOCR

Esta alternativa consiste en utilizar la herramienta MMOCR con el método de reconocimiento + detección de textos para transformar los textos que se encuentran en etiquetas en lenguaje de maquinas.

Las ventajas de esta alternativa son:

- Efectividad: Esta herramienta al ser tan utilizada y testeada se puede ver la gran efectividad que tiene. Los puntos que podrían mermar esta efectividad son los envases de vidrios (difícil de ver los textos), productos que queden en las esquinas de las fotos (productos pueden quedar cortados), reflejos en las fotos, etc.
- Robustez: Al unir la detección y el reconocimiento de textos esta herramienta puede ayudar mucho al momento de transformar las imágenes.
- Generalización: Al haber sido testeado con imágenes tan distintas y entregando una buena efectividad se infiere que la herramienta se adapta bien a distintos tipos de productos, mientras el texto en estos sea legible.

Las desventajas son:

- Tiempo: Para utilizar esta herramienta se necesitan una gran cantidad de prerrequisitos como ya se menciono anteriormente, por lo que instalar todo y entenderlo podría tardar un buen tiempo. Además, se necesita leer mucha documentación antes de empezar a utilizarlo, debido a que consta de una gran cantidad de modelos tanto de reconocimiento como de detección.

Ahora se verán las ventajas y desventajas de los métodos para tratar la salida del OCR:

### 4.2.2. Alternativa 1: Distancia de Levenshtein

Esta alternativa consiste en un método para obtener el número de operaciones necesarias para transformar una cadena de caracteres en otra.

Las ventajas de esta alternativa son:

- Efectividad: Al ser un método que solo cuantifica algo su efectividad es del 100 %, debido a que para cualquier cadena será capaz de devolver un valor.
- Tiempo: Al ser un método conocido y que es tan solo realizar una operación la implementación de este es bastante sencilla y rápida.
- Generalización: Como ya se explico anteriormente este método debería funcionar ante cualquier palabra que le llegué.

Las desventajas son:

- Robustez: Al ser un método que solo cuantifica necesita juntarse con otro método para realizar un cambio en las cadenas.

### 4.2.3. Alternativa 2: Distancia de Levenshtein con peso

Esta alternativa consiste en un método para obtener el número de operaciones necesarias para transformar una cadena de caracteres en otra, pero considerando que no todos los cambios valen lo mismo. Los cambios entre caracteres que se parecen tienen un menor costo en comparación a caracteres que no se parecen.

Las ventajas de esta alternativa son:

- Efectividad: Al ser un método que solo cuantifica algo su efectividad es del 100 %, debido a que para cualquier cadena será capaz de devolver un valor.
- Tiempo: Al ser un método conocido y que es tan solo realizar una operación la implementación de este es bastante rápida. Como es de esperar es un poco menos rápida que la alternativa antes mencionada, debido al tiempo que tardará encontrar unos valores para los pesos que mejoren el método principal.
- Generalización: Como ya se explico anteriormente este método debería funcionar ante cualquier palabra que le llegué.

Las desventajas son:

- Robustez: Al ser un método que solo cuantifica necesita juntarse con otro método para realizar un cambio en las cadenas. Aún así, debería mejorar la robustez de la alternativa 1.

### 4.2.4. Alternativa 3: Hash de Anagramas (Anagram Hashing)

Esta alternativa consiste en un método que sirve para producir un número lo suficientemente grande para poder identificar palabras que tengan los mismos caracteres en común.

Las ventajas de esta alternativa son:

- Tiempo: Al ser un método conocido y que es tan solo realizar una operación la implementación de este es bastante rápida.
- Generalización: Como ya se explicó anteriormente este método debería funcionar ante cualquier palabra que le llegué.

Las desventajas son:

- Efectividad: Al ser un sistema de puntajes que cuantifica igual las palabras que tienen las mismas letras independiente del orden en que estas estén podría afectar mucho en el momento de detectar una palabra, reduciendo así la efectividad de la solución.
- Robustez: Al ser un método que solo cuantifica necesita juntarse con otro método para realizar un cambio en las cadenas.

### 4.2.5. Alternativa 4: Clave de similitud (Similarity key)

El método de clave de similitud lo que busca es mapear palabras a claves con el objetivo de que las palabras que son similares tengan una clave idéntica o por lo menos similar. Este método depende mucho del procesador OCR utilizado y del corpus de datos con el que se trabaja, ya que de esto depende la manera de formar las clases de equivalencia.

Las ventajas de esta alternativa son:

- Tiempo: La máxima dificultad que puede tener este método es la creación de las clases de equivalencia, pero quitando eso no debería tener más inconveniente que provoquen una mayor duración en la implementación de este método.
- Robustez: Al poder poseer todos los caracteres que uno quiera en las clases de equivalencia hace que al momento de equivocarse en el reconocimiento de un caracter, la cadena pueda ser detectada igualmente.

Las desventajas son:

- Efectividad: Al poder tener tantas opciones en las clases de equivalencia puede afectar al momento de detectar la palabra que realmente se quiere encontrar.
- Generalización: Al ser un método en el cual se deben definir unas clases de equivalencia previamente, podría darse el caso de que aparezca una nueva confusión entre caracteres que no haya sido considerada en un principio.

## 4.3. Características de comparación.

Luego de haber definido los puntos a tratar se presenta la siguiente tabla comparativa de las alternativas a partir de las características descritas anteriormente.

| Método | Tiempo | Robustez | Efectividad | Generalización |
|---|---|---|---|---|
| Distancia de Levenshtein | Baja | Baja | Alta | Alta |
| Distancia de Levenshtein con peso | Media-baja | Media-baja | Alta | Alta |
| Hash de Anagramas | Baja | Media-baja | Media | Alta |
| Clave de similitud | Media-baja | Media-Alta | Media | Media |

**Tabla 4.4:** Características comparación

# 5 | Propuesta de solución

## 5.1. Importancia de las características de comparación para las soluciones

A continuación se explica la importancia de cada uno de los métodos en base a las características planteadas en la sección 4.2. El primero y más fundamental para considerar la solución al problema planteado como correcta es la efectividad del método seleccionado, debido a que se espera obtener un sistema funcional que cumpla ciertos estándares al momento de detectar productos, un 75 % de aciertos para ser más específicos. La segunda característica que se le asignó una mayor importancia es la robustez, debido a que se espera que el sistema reciba palabras nuevas constantemente, por lo que tiene que saber como afrontar esta situación de la mejor manera posible. La tercera característica con una mayor importancia es la generalización, la cual tiene un importancia considerablemente menor a las otras dos. Sin embargo, no significa que no deba ser igual de considerada, ya que mientras la solución cumpla una mayor generalización se verá reflejado en el valor de la efectividad, mejorando así la solución seleccionada. Por último, se tiene el tiempo, el cual se le asigna el menor valor de importancia debido a que es la única de las características que no afecta directamente en la efectividad de la solución.

A partir de las ventajas y desventajas presentadas para cada uno de los métodos y la recopilación hecha en la Tabla 4.4 se procede a analizar cada uno de los métodos en cuanto a la importancia dada a cada una de las características.

Primero descartamos el utilizar la distancia de Levenshtein ya que su versión con pesos tiene, en la teoría, igual o mejor rendimiento en cada una de las características planteadas. Como se dijo anteriormente la efectividad es la característica más decisiva a la hora de elegir un método por lo que se considera la distancia de Levenshtein con peso como el método con un mayor potencial. Pero, este método tiene un problema y es que en cuanto a la segunda característica con un mayor potencial, la robustez, tiene un bajo potencial. Es por esto que se considerará incluir un segundo método y es que la clave de similitud cuenta con el mejor potencial en cuanto a la robustez- Por ende se descarta utilizar la solución del hash de anagramas ya que no supera en las características con mayor importancia a ninguno de los otros métodos.

## 5.2. Mejor alternativa según su conjunto de características

A partir de los argumentos presentados en la sección 5.1 podemos notar que las alternativa que tiene un mayor potencial en la característica con mayor importancia es la distancia de Levenshtein

con peso, pero se descompensa en la robustez. Es por esto que para suplir esta falta se opta como solución al problema utilizar dos métodos en conjunto. Este segundo método será la clave de similitud. Esta idea viene a partir de notar que prácticamente las ventajas de uno de los métodos son las desventajas del otro y viceversa.

Para esta alternativa se considera que cada método supla las carencias del otro, por lo que en los valores de las características se considerará el mejor entre ambos métodos.

| Método | Tiempo | Robustez | Efectividad | Generalización |
|---|---|---|---|---|
| Distancia de Levenshtein con peso + Clave de similitud | Media-baja | Media-Alta | Alta | Alta |

**Tabla 5.1:** Porcentajes de la mejor solución

Como se puede notar realizando esta combinación de métodos se gana mucho en cuanto a la robustez de la solución del problema, lo cual representa una gran mejora. Cabe recalcar que esto es posible debido a que no son excluyentes estos métodos entre si.

Finalmente se concluye que la mejor opción es la de mezclar el método de Distancia de Levenshtein con peso y el método de Clave de similitud. Cabe recalcar que de todos modos se probarán las distintas formas en que la distancia de Levenshtein podría ayudar a solucionar el problema.

# 6 | Desarrollo de la solución.

## 6.1. Desarrollo de las etapas planteadas

### 6.1.1. Elegir el modelo de reconocimiento que se utilizará

La página de MMOCR provee una serie de modelos tanto de detección de texto como de reconocimiento de texto los cuales fueron mencionados en la sección 4.1.1.1 y 4.1.1.2. Para la detección se utilizará el modelo base que viene, sin entrar en mucho más detalle de la arquitectura que posee. Esto es debido a que no se utilizará la información que nos entrega en detalle. Caso contrario es el del modelo de reconocimiento que se utilizará, debido a que el que se elija influirá directamente en la calidad de las palabras reconocidas. Tener una mejor calidad en las palabras que se reconocen en una imagen influye directamente al momento de saber de que producto se trata.

Los modelos de reconocimiento disponibles vienen ya entrenados y evaluados en distintos sets tanto de texto regular como otros de texto irregular. Entre todos los modelos que se entregan el que más destaca en principio es el llamado ABINet. Si comparamos sus resultado con los de otros dos modelos notamos una amplia diferencia en los porcentajes de precisión obtenidos. Estas estadísticas están sacadas a partir tres datasets de texto regular (IIIT5K, SVT y IC13) y tres datasets de texto irregular (IC15, SVTP y CT80) y los resultados se pueden observar de mejor manera en la Tabla 6.1

| Método | Texto regular | Texto regular | Texto regular | Texto irregular | Texto irregular | Texto irregular |
|---|---|---|---|---|---|---|
| - | IIIT5K | SVT | IC13 | IC15 | SVTP | CT80 |
| ABINet | 95.7 % | 94.6 % | 95.7 % | 85.1 % | 90.4 % | 90.3 % |
| CRNN | 80.5 % | 81.5 % | 86.5 % | 54.1 % | 59.1 % | 55.6 % |
| SEG | 90.9 % | 81.8 % | 90.7 % | - | - | 80.9 % |

**Tabla 6.1:** Comparación entre los modelos de reconocimiento.

A pesar de que los resultados pueden parecer abrumadores a simple vista, los textos que existen en los distintos productos del retail podrían llegar a ser muy distintos a los que se posee en estos datasets. Es por esta razón que ahora se evaluarán estos modelos, pero con imágenes reales que deberá analizar la solución final. El dataset que se utilizará contiene un total de 569 imágenes de distintos productos separados en 9 categorías, las cuales son:

- Afeitado: Contiene un total de 72 imágenes de distintos productos como máquinas de afeitar, espumas, etc.
- Bebidas y aguas: Contiene un total de 67 imágenes de distintos productos como bebidas energéticas, aguas minerales, leches, etc.
- Coloración: Contiene un total de 26 imágenes de distintos tipos de tinturas, donde la mayor dificultad es que en muchos casos la única diferencia es solo el color de la tintura o un simple número.
- Cremas corporales: Contiene un total de 73 imágenes de distintos tipos de cremas.
- Desodorantes corporales: Contiene un total de 194 imágenes de tipos de desodorantes, tanto femeninos como masculinos.
- Fragancias: Contiene un total de 39 imágenes de distintos tipos de fragancias.
- Pañales desechables: Contiene un total de 40 imágenes de distintos tipos de pañales de bebés.
- Protección sanitaria: Contiene un total de 13 imágenes de distintos productos como protectores sanitarios, jabones íntimos, etc.
- Tocador infantil: Contiene un total de 45 imágenes de distintos productos como colonias, jabones, shampoo etc. Todo estos productos enfocados a niños menores.

La manera de evaluar si los modelos detectan correctamente o no el producto que aparece en la imagen es la siguiente:

1. Se pasará a cada uno de los modelos todo el dataset de imágenes y se obtendrá una predicción de las palabras que poseen.
2. Se cargará la base de datos con las palabras que contiene cada producto.
3. Se utilizará una función que, a través de un sistema de puntajes, se asocie la predicción de las palabras reconocidas con cada una de las palabras que se encuentren en la base de datos. Esta función retornará una pequeña lista con las posibilidades de productos detectadas. Este se especifica en el Anexo A.1.
4. Se calculará una estadística que considere si el producto real se encuentra dentro de las posibilidades detectadas en la función antes mencionada.

A partir de realizar este proceso para los distintos modelos es que se obtienen los resultados observados en la Tabla 6.2

| Categorías | CRNN | SEG | ABINet |
|---|---|---|---|
| Afeitado | 53 % | 58 % | 71 % |
| Bebidas y aguas | 39 % | 57 % | 72 % |
| Coloración | 62 % | 73 % | 85 % |
| Cremas corporales | 63 % | 67 % | 71 % |
| Desodorantes corporales | 58 % | 55 % | 72 % |
| Fragancias | 54 % | 67 % | 77 % |
| Pañales desechables | 53 % | 65 % | 78 % |
| Protección sanitaria | 62 % | 85 % | 92 % |
| Tocador infantil | 51 % | 71 % | 78 % |
| Total | 55 % | 62 % | 74 % |

**Tabla 6.2:** Comparación entre los modelos de reconocimiento en datos de productos reales.

Los cuales se pueden visualizar de mejor manera en la Figura 6.1

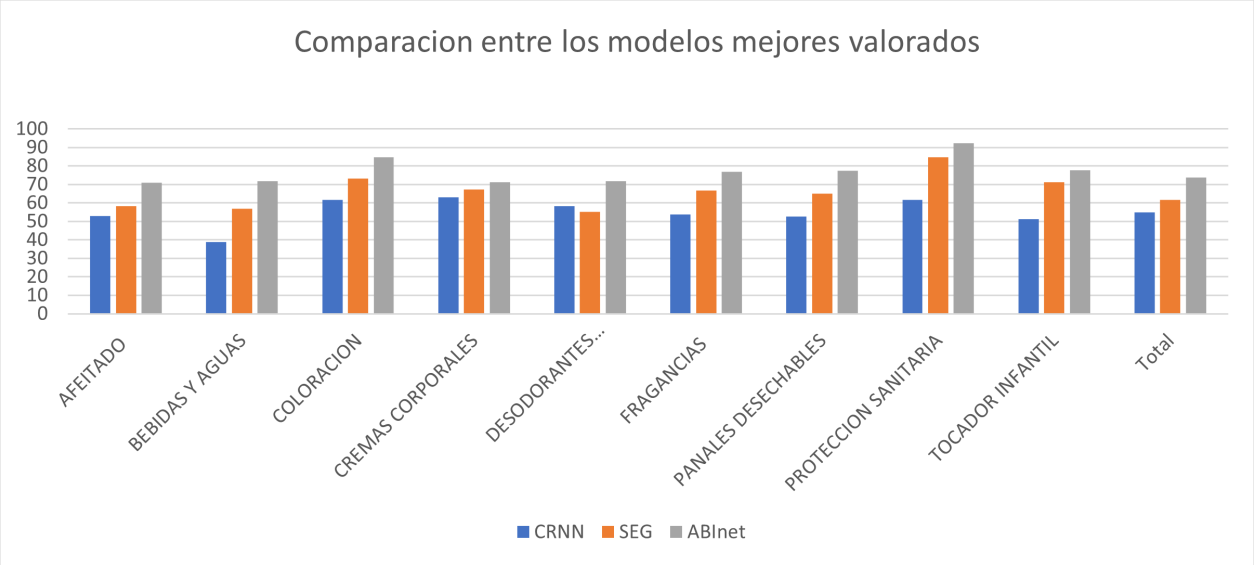



**Figura 6.1:** Gráfica de la comparación entre modelos de reconocimiento aplicado a imágenes reales.

A partir de esto se puede observar una clara superioridad del modelo ABINet en la calidad del reconocimiento de caracteres. Esto se puede justificar debido a sus grandes números de precisión en textos irregulares, el cual se ve presente en varias marcas de productos.

Ahora se entrará más en profundo en como funciona el modelo seleccionado. El modelo ABINet (1) obtiene su nombre a partir de 3 conceptos en los que se basa. En primer lugar, se tiene la letra 'A' que viene del principio autónomo. Este principio aplicado al reconocimiento de texto en escena (STR por sus siglas en inglés) significa que los modelos de reconocimiento deben desacoplarse en modelo de visión (VM por sus siglas en inglés) y modelo de lenguaje (LM por sus siglas en inglés), y los submodelos pueden servir como unidades funcionales de forma independiente y aprender por separado. En segundo lugar se tiene la letra 'B' que viene del principio bidireccional. La acción de razonar el contexto del caracter se comporta como una cloze test ya que los caracteres ilegibles pueden verse como espacios en blanco. Por lo tanto, la predicción se puede hacer utilizando las señales de caracteres legibles que se tenga en el lado izquierdo y derecho de los caracteres

ilegibles simultáneamente. En tercer lugar se tiene la letra 'I' que viene del principio iterativo. Este principio se aplica en el modelo de lenguaje y se utiliza para refinar la predicción a partir de señales visuales y lingüísticas. Al introducir los resultados de ABINet en LM repetidamente, las predicciones se pueden refinar progresivamente generando así una solución basado en el aprendizaje semisupervisado. Este modelo se ve representado en la Figura 6.2

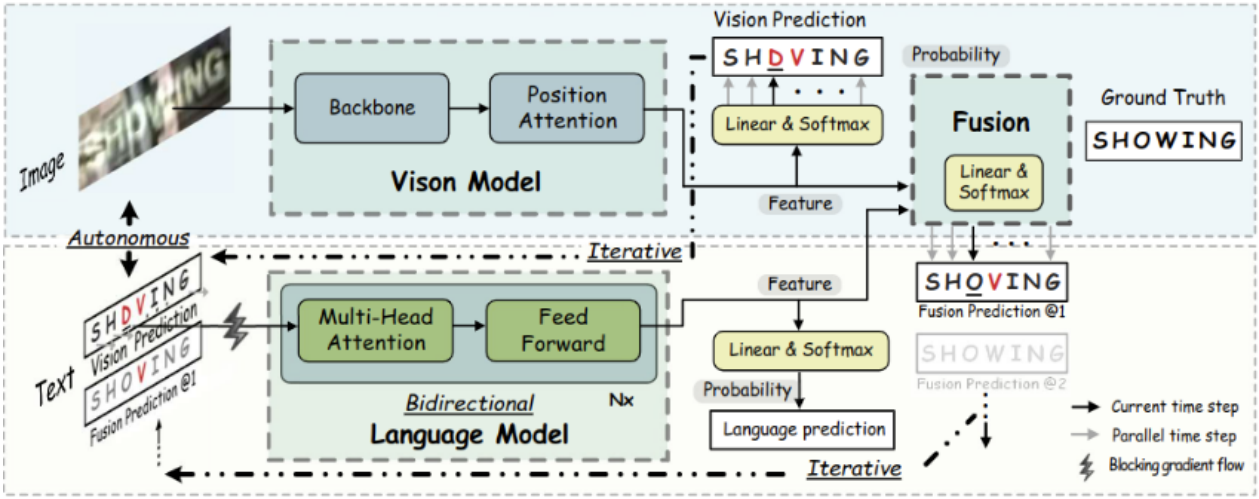



**Figura 6.2:** Arquitectura del modelo ABINet (1).

En la figura se observa un modelo desacoplado al bloquear el flujo del gradiente entre el VM y el LM, lo que obliga a LM a aprender explícitamente reglas lingüísticas. Además, tanto VM como LM son unidades autónomas y se pueden entrenar previamente a partir de imágenes y texto por separado. Como entrada del modelo se tiene una imagen, la cual entra en el modelo de visión para así obtener la primera predicción del texto que posee la imagen. Luego el texto que predijo el VM entra al LM de manera iterativa para así analizar la composición del texto encontrado. A partir de esto ingresa en un modelo de fusion, donde analiza tanto la salida del VM y el LM para así realizar las correcciones pertinentes. Por último, la salida de la predicción corregida vuelve a ingresar al LM de manera iterativa para así refinar la precisión de la predicción lo máximo posible.

### 6.1.2. Crear clases de equivalencia

Las clases de equivalencia corresponde a una serie de listas que contienen los caracteres que son similares entre sí. Por ejemplo, los caracteres 'e' y 'c' entran en una misma clase de equivalencia debido a su forma.

A partir de un análisis hecho de los caracteres que se encuentran en las etiquetas de productos y los caracteres que detecta el modelo se formaron las siguientes clases:

- La primera se crea a partir de los caracteres que tengan una forma alargada en su zona central. Estos corresponden a ['f','i','j','l','r','t','1'].
- La segunda se crea a partir de caracteres que tengan una forma circular en cualquier zona de este. Estos corresponden a ['a','b','d','g','o','p','q','0'].
- La tercera se crea a partir de caracteres que tengan algún tipo de forma de 'u'. Estos corresponden a ['v','w','y','u'].

- Las siguientes clases se forman a partir de similitudes más específicas entre los caracteres. Ahora, se mostrarán los pares de caracteres más similares entre si. Estos corresponden a ['e','c'], ['s','5'] y ['h','k'].

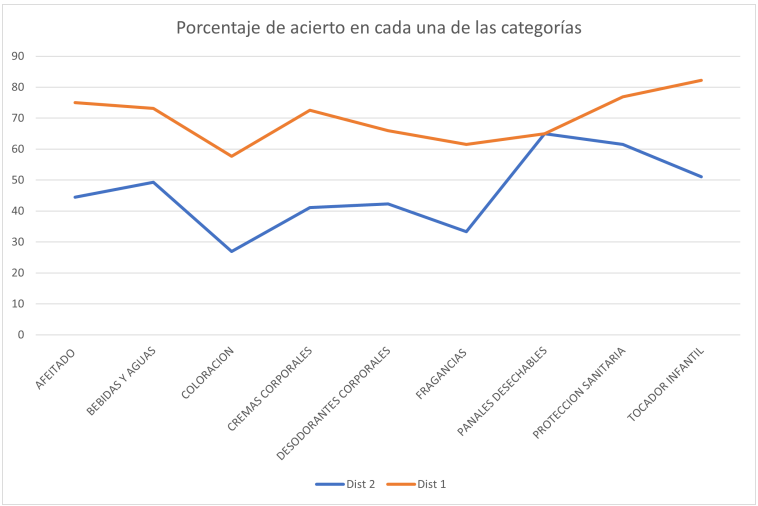
### 6.1.3. Implementar lo métodos de corrección de errores del OCR

#### 6.1.3.1. Distancias de Levenshtein 1 y 2

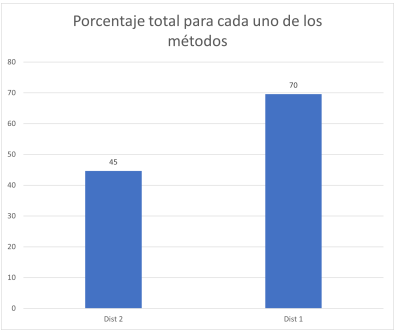
Como se especificó en la sección 4.1.2 y 4.1.3 la distancia de Levenshtein consiste en un método para contabilizar la cantidad de cambios esenciales para llegar de una cadena de caracteres a otra. La primera idea que se tomó fue implementar una función que a partir del método de la distancia de Levenshtein encontrara todas las cadenas de caracteres que se encuentran a una distancia de 1. Este método se implementa en el Anexo A.2. Para utilizarlo lo que se hizo fue aplicar esta función sobre cada una de las cadenas de caracteres detectadas por el modelo de reconocimiento. Luego verificar si en alguna de las cadenas a distancia 1 se encontraba una posible palabra. Para verificar esto se reviso el diccionario que contiene todas las posibles palabras. Si es que en la lista de palabras a distancia 1 se detectó más de una palabra que se encuentra en el diccionario de todas las palabras se decidirá con la palabra que tenga una mayor frecuencia en todo el corpus de datos.

Para dar un margen más amplio y probar nuevos métodos se optó por utilizar la misma idea planteada anteriormente, pero ahora se incluirán todas las palabras que se encuentren a una distancia de Levenshtein de 2. La función que nos entrega estas palabras se muestra en el Anexo A.3. A partir de utilizar estos dos métodos se llegaron a los siguientes resultados.

En la Figura 6.3(a) se puede observar como el método de la distancia de Levenshtein de 1 supera ampliamente en casi todas las categorías a su similar de distancia 2 dando el porcentaje final que se observa en la Figura 6.3(b)




((a)) Comparación entre el método de Dist de Lev 1 y Dist de Lev 2 por categoría




((b)) Comparación entre el método de Dist de Lev 1 y Dist de Lev 2 en total

**Figura 6.3:** Comparación entre el método de Dist de Lev 1 y Dist de Lev 2

A partir de los valores obtenidos en la Figura 6.3(b) es que se nota que los resultados no llegan a los valores mínimos que requiere el sistema para considerarse que cumple con lo solicitado. Estos

valores fueron calculados a partir de considerar el mejor resultado al aplicar el módulo de asociación con la base de datos. Ahora recordamos que el objetivo es que se encuentre el producto real dentro de las posibilidades reconocidas por este módulo. Es por esto que se propone aumentar el rango de diferencia entre el producto que obtiene un mayor porcentaje y el resto de productos. Se harán tres pruebas: una con diferencia de 0, otra con diferencia de 1 y otra con una diferencia de 2. Los resultados al realizar esto se ven reflejados en la Tabla 6.3

| Categorías | dif = 0 | dif = 1 | dif = 2 |
|---|---|---|---|
| Afeitado | 75 % | 78 % | 93 % |
| Bebidas y aguas | 73 % | 87 % | 91 % |
| Coloración | 58 % | 88 % | 92 % |
| Cremas corporales | 73 % | 90 % | 93 % |
| Desodorantes corporales | 66 % | 80 % | 88 % |
| Fragancias | 62 % | 72 % | 82 % |
| Pañales desechables | 65 % | 78 % | 88 % |
| Protección sanitaria | 77 % | 85 % | 92 % |
| Tocador infantil | 82 % | 87 % | 93 % |
| Total | 70 % | 82 % | 90 % |

**Tabla 6.3:** Comparación entre el mismo modelo pero variando la diferencia.

Considerando como aumentan los porcentajes parecería a simple vista que la opción de una diferencia de 2 hay que elegirla siempre, pero hay una consideración que hay que tener. Aumentar ese valor hará que se considere una mayor cantidad de posibles productos, lo cual si esta cantidad se vuelve muy grande, podría dificultar un posible futuro modelo que busque relaciones entre estos productos y la imagen original. Es por esto que en la Figura 6.4 se presenta como aumentan estos valores a medida que aumenta el valor de la diferencia.

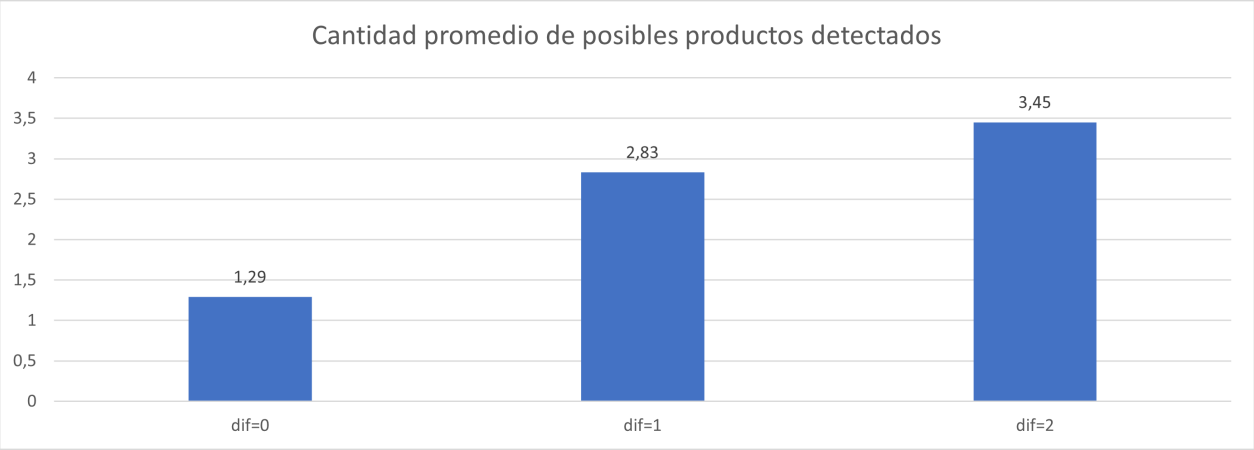



**Figura 6.4:** Cantidad promedio de posibles productos detectados

La manera de interpretar estos datos es la siguiente: Para el método de la distancia de Levenshtein 1, si es que consideramos una diferencia de 2 entre el mayor puntaje obtenido y el resto de puntajes, se obtiene que en el 90 % de los casos se encuentra el producto dentro de 3,5 posibles productos. Tener en promedio 3,5 posibles productos para cada imagen de entrada es un valor bastante razonable y dentro de los márgenes que se consideran en un principio, por lo que esta parece una opción viable a implementar como propuesta final.

#### 6.1.3.2. Considerar la distancia de Levenshtein menor

Siguiendo con la idea de utilizar la distancia de Levenshtein como un algoritmo para encontrar errores en predicciones de los modelos OCR es que se vuelve a la idea original. Se utilizará el algoritmo para cada una de las palabras detectadas y se calculará la distancia a cada una de las palabras que se encuentran en el diccionario que almacena todas las palabras de la base de datos. El algoritmo está descrito en el Anexo A.4. A partir de esto se escogerá la palabra que tenga una menor distancia como la posible palabra reconocida por el modelo. Por consiguiente de utilizar este método es que se obtienen los porcentajes de detección mostrados en la Tabla 6.4

| Categorías | dif = 0 | dif = 1 | dif = 2 |
|---|---|---|---|
| Afeitado | 89 % | 94 % | 97 % |
| Bebidas y aguas | 73 % | 90 % | 96 % |
| Coloración | 77 % | 96 % | 96 % |
| Cremas corporales | 78 % | 93 % | 97 % |
| Desodorantes corporales | 75 % | 88 % | 94 % |
| Fragancias | 77 % | 79 % | 82 % |
| Pañales desechables | 78 % | 90 % | 93 % |
| Protección sanitaria | 85 % | 92 % | 100 % |
| Tocador infantil | 87 % | 89 % | 96 % |
| Total | 78 % | 90 % | 94 % |

**Tabla 6.4:** Comparación utilizando la distancia Levenshtein más próxima pero variando la diferencia.

Como se puede observar utilizando una diferencia de 0 se obtiene un total de reconocimientos del 78 % con un promedio de 1.1 productos en las posibilidades como se puede observar en la Figura 6.5. Esto significa que con este método se puede obtener un porcentaje de detección de productos mayor al 75 % y con prácticamente una solo posibilidad entre las devueltas por el sistema. De todos modos sigue siendo viable el aplicar la diferencia de 2 y obtener un porcentaje de 94 % en las detecciones con menos de 4 productos entre las posibilidades.

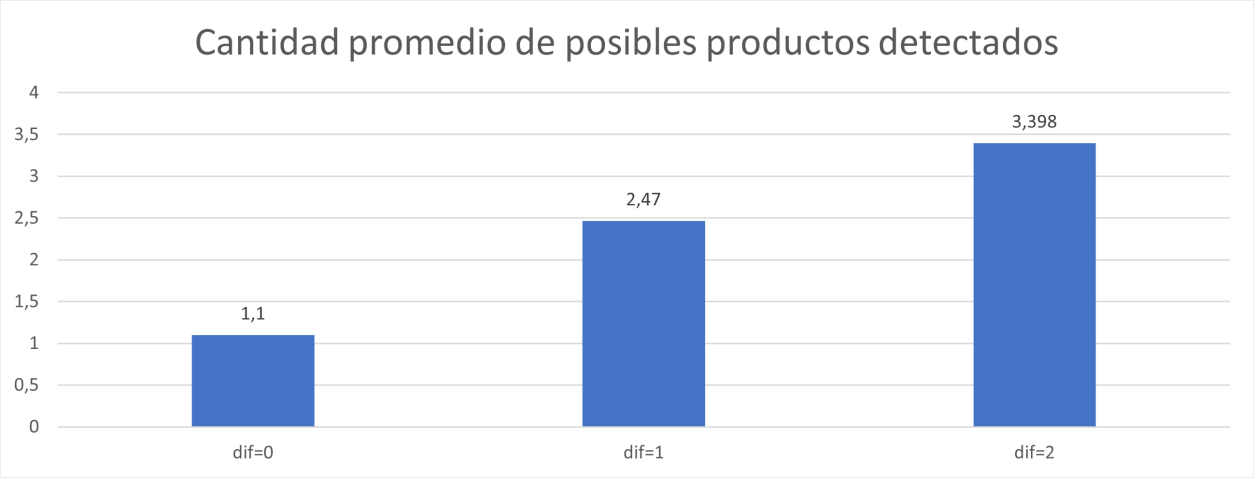



**Figura 6.5:** Cantidad promedio de posibles productos detectados para el método de distancia de Levenshtein más próxima

#### 6.1.3.3. Considerar la distancia de Levenshtein menor y utilizar claves de similitud

Por último, se tiene el método mejor valorado en la teoría que es el de utilizar la distancia de Levenshtein junto a pesos que se definen en base a las clases de equivalencia definidas en la sección 6.1.2. Para definir los pesos se optó por la siguiente metodología: Para las clases mencionadas en los primeros 3 puntos de la sección 6.1.2 se le dio un valor de 0,5 a un cambio esencial entre dos caracteres que correspondan a la misma clase y para las clases definidas en el último punto se consideró un valor de 0,25 debido a que la similitud entre estas clases se consideró mayor a la similitud que existe entre las otras clases. A partir de aplicar este método en las palabras detectadas por el modelo ABINet es que se obtienen los resultados presentados en la Tabla 6.5

| Categorías | dif = 0 | dif = 1 | dif = 2 |
|---|---|---|---|
| Afeitado | 88 % | 94 % | 97 % |
| Bebidas y aguas | 73 % | 90 % | 94 % |
| Coloración | 77 % | 96 % | 96 % |
| Cremas corporales | 78 % | 93 % | 97 % |
| Desodorantes corporales | 76 % | 89 % | 94 % |
| Fragancias | 77 % | 79 % | 82 % |
| Pañales desechables | 78 % | 90 % | 93 % |
| Protección sanitaria | 92 % | 92 % | 100 % |
| Tocador infantil | 87 % | 89 % | 96 % |
| Total | 79 % | 91 % | 95 % |

**Tabla 6.5:** Comparación utilizando la distancia Levenshtein más próxima pero variando la diferencia.

A partir de esta tabla se puede observar que utilizando una diferencia de 0 se obtiene un total de reconocimientos del 79 % con un promedio de 1,162 productos en las posibilidades como se observa en la Figura 6.6. Esto significa que con este método, al igual que con el anterior, se puede obtener un porcentaje de detección de productos mayor al 75 % y con prácticamente una solo posibilidad entre las devueltas por el sistema. De todos modos sigue siendo viable el aplicar la diferencia de 2 y obtener un porcentaje de 95 % en las detecciones con menos de 4 productos entre las posibilidades.

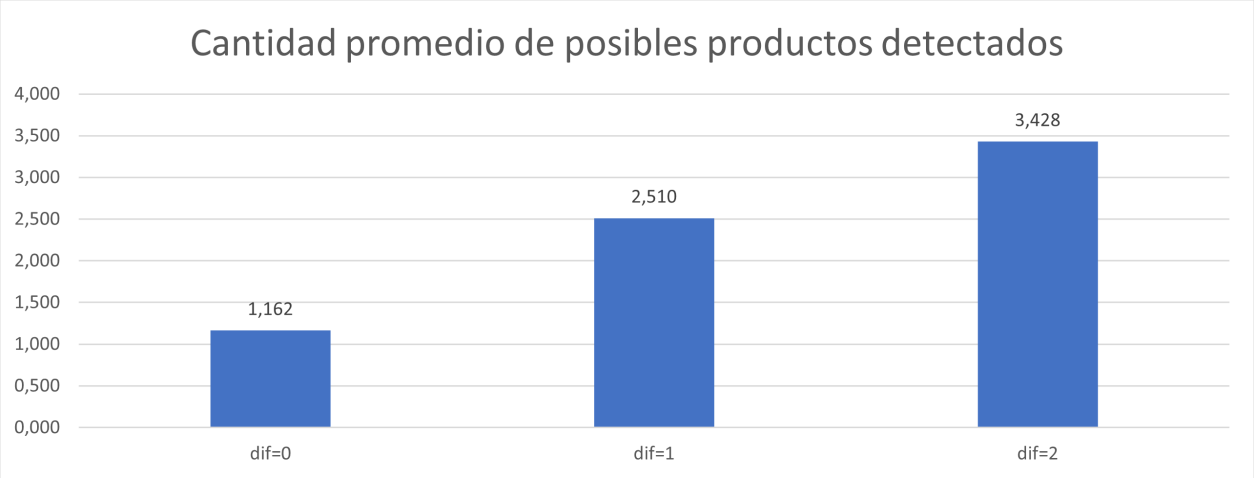



**Figura 6.6:** Cantidad promedio de posibles productos detectados para el método de distancia de Levenshtein más próxima con clave de similitud

# 7 | Conclusiones y posibles trabajos a futuro

## 7.1. Conclusiones

Después de todas las estadísticas presentadas y todo el trabajo realizado parecería lógico escoger simplemente el método con un mayor porcentaje de aciertos en las posibilidades entregadas, pero queda un objetivo de los planteados por revisar. El tiempo de ejecución que se demora cada método, incluyendo lo que tarda el modelo ABINet en el reconocimiento del texto en las imágenes no puede superar los 5 segundos por imagen. Primero se presenta la Tabla 7.1 donde se muestran los tiempos de ejecución de cada uno de los modelos, tanto para todo el set como también se presenta el tiempo promedio por cada una de las imágenes.

| Modelo | Tiempo [s] (set) | tiempo [s] (por imagen) |
|---|---|---|
| CRNN | 494 | 0,868189807 |
| SEG | 988 | 1,736379613 |
| ABINet | 845 | 1,485061511 |

**Tabla 7.1:** Comparación de tiempos de ejecución entre los distintos modelos

Como ya se comprobó que el mejor modelo a utilizar es ABINet, independiente de que tarde más en su ejecución, se cuenta con un tiempo de ejecución máximo para el método a seleccionar de 3,5 segundos aproximadamente.

Ahora se presentan los tiempos de ejecución que tarda cada uno de los métodos en la Tabla 7.2.

| Método | Tiempo [s] (set) | tiempo [s] (por imagen) |
|---|---|---|
| Distancia de Levenshtein de 1 | 1,2 | 0,002108963 |
| Distancia de Levenshtein de 2 | 307 | 0,539543058 |
| Distancia de Levenshtein más próxima | 984 | 1,729349736 |
| Distancia de Levenshtein más próxima con clave de similitud | 1687 | 2,964850615 |

**Tabla 7.2:** Comparación de tiempos de ejecución entre los distintos métodos

A partir de toda la información recopilada es que se llega a las siguientes conclusiones para cada uno de los métodos:

- Para el método de la distancia de Levenshtein de 2 se tiene un tiempo que cumple con bastante creces los objetivos planteados y, aunque el porcentaje de detección puede alcanzar los valores

planteados en los objetivos si es que se utiliza una diferencia de 2, queda corto ante los valores que alcanzan los otros métodos.

- Para el método de distancia de Levenshtein más próxima con clave de similitud es el que obtiene un mayor porcentaje de aciertos, pero a su vez alcanza el mayor valor de tiempo de ejecución. También cabe mencionar que el aumento en el porcentaje de detección no es lo suficientemente considerable como para suplir el aumento en el tiempo de ejecución con respecto a los otros métodos.
- Para el método de la distancia de Levenshtein de 1 se tiene el mejor tiempo de ejecución por imagen con una amplia diferencia con respecto a los otros. Además, el porcentaje de detección es de aproximadamente un 90 %, por lo que este método tiene la mejor relación entre porcentaje de detección y el tiempo de ejecución que este tarda.
- Para el método de distancia de Levenshtein más próxima se tiene un aumento que ya es considerable en el porcentaje de detección con respecto al método anterior al pasar de un 90 % a un 94 %, pero, a su vez, el tiempo de ejecución tiene un aumento en 3 ordenes de magnitud. De todos modos el tiempo de ejecución de este método sigue estando dentro de los márgenes establecidos y deja una holgura para un posible post procesamiento.

## 7.2. Posibles trabajos a futuro

Como se planteó desde un principio la idea principal de este trabajo es, a través de un sistema, obtener una lista acotada de posibles productos a través de ingresar una imagen de este. Pero la lista de posibles productos siempre se puede acotar más hasta llegar a un solo producto que sea el definitivo. Es por esto que como post procesamiento se plantea utilizar un modelo que reconozca diferencias entre imágenes de la forma más minuciosa posible. Para desarrollar esta idea lo único que se necesitaría tener es una base de datos con imágenes de referencia de cada uno de los productos posibles. A partir de esta idea es que si se llega a utilizar un modelo con un tiempo de ejecución que tarde menos de 1,5 segundos (aproximadamente) es viable utilizar el método distancia de Levenshtein más próxima, pero si el modelo tarda más de 1,5 segundos se tendría que optar por es coger el método de distancia de Levenshtein de 1 al ser la propuesta que tiene la mejor relación entre tiempo de ejecución y porcentaje de productos detectados.

# Bibliografía

[1] S. Fang, H. Xie, Y. Wang, Z. Mao, and Y. Zhang, "Read like humans: Autonomous, bidirectional and iterative language modeling for scene text recognition," 2021. (document), **??**, 6.1.1, 6.2

[2] Z. Kuang, H. Sun, Z. Li, X. Yue, T. H. Lin, J. Chen, H. Wei, Y. Zhu, T. Gao, W. Zhang, K. Chen, W. Zhang, and L. Dahua, "Mmocr: A comprehensive toolbox for text detection, recognition and understanding," 2021. 2.2.1, 4.1.1

[3] N. Kanellakis, C. and G. Survey, "Survey on computer vision for uavs: Current developments and trends," 2017. 3.1

[4] K. B. Amarjot Singh and A. Bhasin, "A survey of ocr applications," 2012. 3.2

[5] "Magro, r. (2013). binarización de imágenes digitales y su algoritmia como herramienta aplicada a la ilustración entomológica. boletín de la sociedad entomológica aragonesa, 53, 443-464," 2013. 3.2

[6] P. A. Paliza, "Corrección automática de errores de ocr en documentos semi-estructurados (bachelor's thesis)." 2016. 3.2.2

[7] M. Liao, Z. Wan, C. Yao, K. Chen, and X. Bai, "Real-time scene text detection with differentiable binarization," *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 11 474–11 481, 2020. **??**, **??**

[8] M. Liao, Z. Zou, Z. Wan, C. Yao, and X. Bai, "Real-time scene text detection with differentiable binarization and adaptive scale fusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. **??**

[9] S.-X. Zhang, X. Zhu, J.-B. Hou, C. Liu, C. Yang, H. Wang, and X.-C. Yin, "Deep relational reasoning graph network for arbitrary shape text detection," pp. 9699–9708, 2020. **??**

[10] Y. Zhu, J. Chen, L. Liang, Z. Kuang, L. Jin, and W. Zhang, "Fourier contour embedding for arbitrary-shaped text detection," in *CVPR*, 2021. **??**

[11] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988. **??**

[12] W. Wang, E. Xie, X. Song, Y. Zang, W. Wang, T. Lu, G. Yu, and C. Shen, "Efficient and accurate arbitrary-shaped text detection with pixel aggregation network," in *ICCV*, 2019, pp. 8439–8448. **??**

[13] W. Wang, E. Xie, X. Li, W. Hou, T. Lu, G. Yu, and S. Shao, "Shape robust text detection with progressive scale expansion network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9336–9345. **??**

[14] S. Long, J. Ruan, W. Zhang, X. He, W. Wu, and C. Yao, "Textsnake: A flexible representation for detecting text of arbitrary shapes," pp. 20–36, 2018. **??**

[15] X. Q. Y. C. P. G. R. X. X. B. Ning Lua, Wenwen Yua, "Master: Multi-aspect non-local network for scene text recognition," 2021. 4.1.1.2

[16] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE transactions on pattern analysis and machine intelligence*, 2016. **??**

[17] N. Lu, W. Yu, X. Qi, Y. Chen, P. Gong, R. Xiao, and X. Bai, "MASTER: Multi-aspect non-local network for scene text recognition," *Pattern Recognition*, 2021. **??**

[18] F. Sheng, Z. Chen, and B. Xu, "Nrtr: A no-recurrence sequence-to-sequence model for scene text recognition," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2019, pp. 781–786. **??**

[19] X. Yue, Z. Kuang, C. Lin, H. Sun, and W. Zhang, "Robustscanner: Dynamically enhancing positional clues for robust text recognition," in *European Conference on Computer Vision*, 2020. **??**

[20] H. Li, P. Wang, C. Shen, and G. Zhang, "Show, attend and read: A simple and strong baseline for irregular text recognition," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 8610–8617. **??**

[21] J. L. S. P. J. B. S. J. O. S. Kim and H. Lee, "On recognizing texts of arbitrary shapes with 2d self-attention," 2019. **??**

[22] B. Shi, X. Wang, P. Lyu, C. Yao, and X. Bai, "Robust scene text recognition with automatic rectification," 2016. **??**

[23] "Segocr simple baseline." 2021, unpublished Manuscript. **??**

[24] R. A. Wagner and M. J. Fischer., "The string-to-string correction problem. journal of the acm (jacm), 21(1):168–173," 1974. 4.1.2

# A | Anexo

## A.1. Módulo de asociación con la base de datos

```
#lista_palabras: lista de palabras para comparar con base de datos.
#categoreia: categoria donde se encuentra el producto.
#diccionario_sku: diccionario con las palabras detectadas por el OCR para cada uno de los productos.
def MACBD (lista_palabras,categoria,diccionario_sku):
    lista_puntajes_sku = [] #lista de listas de la forma [[puntaje_1,sku_1], [puntaje_2,sku_2], ...]
    posibles_skus = diccionario_planograma[categoria] #lista de skus posibles para la categoria
    diferencia = 1 #Maxima diferencia para considerar que el producto entra en la lista de posibles skus
    for posible_sku in posibles_skus:
        if posible_sku in diccionario_sku.keys():
            palabras_de_cada_sku = diccionario_sku[posible_sku]
            puntaje = 0
            for palabra in lista_palabras:
                if (palabra in palabras_de_cada_sku):
                    #Sistema de puntajes
                    indice = palabras_de_cada_sku.index(palabra)
                    if (indice == 0) or (indice == 1):
                        puntaje += 4
                    elif (indice == 2) or (indice == 3):
                        puntaje += 3
                    elif (indice == 4) or (indice == 5):
                        puntaje += 2
                    else:
                        puntaje += 1
            lista_puntajes_sku.append([puntaje, posible_sku])
    lista_puntajes_sku.sort(reverse = True)
    output = []
    try:
        puntaje_maximo = lista_puntajes_sku[0][0]
        if (puntaje_maximo > 0):
            for puntaje in lista_puntajes_sku:
                if(puntaje[0] >= puntaje_maximo - diferencia) and (puntaje[0] <= puntaje_maximo + diferencia):
                    output.append(puntaje[1])
                else:
                    break
    except:
        pass
    return output
```

## A.2. Función para obtener todas las cadenas de caracteres a una distancia de Levenshtein de 1

```
def Lev1(palabra):
    p = 'abcdefghijklmnopqrstuvwxyz'
    splits     = [(palabra[:i], palabra[i:]) for i in range(len(palabra) + 1)]
    deletes    = [a + b[1:] for a, b in splits if b]
    replaces   = [a + c + b[1:] for a, b in splits for c in p if b]
    inserts    = [a + c + b     for a, b in splits for c in p]
    return set(deletes + replaces + inserts)
```

## A.3. Función para obtener todas las cadenas de caracteres a una distancia de Levenshtein de 2

```
def Lev2 (palabra) :
    return (set(e2 for e1 in Lev1(palabra) for e2 in Lev1(e1)))
```

## A.4. Función para obtener la distancia de Levenshtein entre dos cadenas de caracteres

```
def lev(seq1, seq2):
    size_x = len(seq1) + 1
    size_y = len(seq2) + 1
    matrix = np.zeros ((size_x, size_y))
    for x in range(size_x):
        matrix [x, 0] = x
    for y in range(size_y):
        matrix [0, y] = y
    for x in range(1, size_x):
        for y in range(1, size_y):
            if seq1[x-1] == seq2[y-1]:
                matrix [x,y] = min(
                    matrix[x-1, y] + 1,
                    matrix[x-1, y-1],
                    matrix[x, y-1] + 1
                )
            else:
                matrix [x,y] = min(
                    matrix[x-1,y] + 1,
                    matrix[x-1,y-1] + 1,
                    matrix[x,y-1] + 1
                )
    return (int(matrix[size_x - 1, size_y - 1]))
```