Experimentar con los elementos más importantes de Google Lens

David Sánchez González

Resumen– Este proyecto se centrará en investigar sobre los elementos que hacen posible el funcionamiento de Google Lens. Dividiendo el contenido en tres apartados principales, como son la detección, extracción y traducción de texto. Haciendo una investigación para escoger al menos dos de los algoritmos o redes neuronales más adecuados para cada tarea, explicando brevemente el funcionamiento de los mismos. También se escogen un conjunto de datasets de imágenes especialmente pensado para escenas de detección de texto. Tres de ellos serán datasets públicos mencionados en artículos científicos similares, además de generar uno propio que contará con imágenes que diremos exteriores e interiores. Exteriores son principalmente carteles de restaurantes, bares, tiendas y otros establecimientos y señalizado públicos. Mientras que las interiores serán imágenes que se pueden encontrar dentro de estos establecimientos, como menús de restaurantes, libros en una librería u otros objetos que se puedan encontrar incluso dentro de casa que contengan texto. Estos datasets serán utilizados para comparar los algoritmos escogidos anteriormente y determinar cuál de ellos es el más adecuado para cada tarea.

Palabras clave– Google Lens, OCR, Tesseract, Keras, EAST Detector, DeepL, GPT-3, OpenAI, MSER, Detección de texto

Abstract– This project will be focused around investigating the elements that make possible Google Lens. It will be divided in three main areas, text detection, extraction and translation. Researching to pick at least two algorithms or neural networks most adequate for each task, giving a brief explanation of each of them. There'll also be image datasets selected especially for text detection. Three of them will be public datasets previously mentioned in various scientific articles, in addition to those three, a fourth one will be made with my own images, both outdoors and indoors. To clarify, outdoors are, like the name specifies, images taken outside, consisting of images with different signs, like those of a restaurant, bar, other establishments and public road signs. Indoors are referring to images you can find inside those establishments, like a menu inside a restaurant, books on a library or other objects that contain text that can be found inside a house. These datasets will later be used to test and compare the selected algorithms and compare results between them to determine which of them provides better results.

Keywords– Google Lens, OCR, Tesseract, Keras, EAST Detector, DeepL, GPT-3, OpenAI, MSER, Text detection

1 Introducción

VIVIMOS en un mundo en el que la tecnología coge un papel cada vez más relevante, a la vez que avanza más rápido de lo que podríamos haber anticipado, pudiendo llevar en nuestro bolsillo la potencia de un ordenador, con la capacidad de acceder a un mundo completo de información casi al instante. Desde 2017, estos pequeños ordenadores también tiene la capacidad de detectar, leer en voz alta e incluso traducir textos de otros idiomas solo con apuntarlo con la cámara de tu *smartphone* haciendo uso de la aplicación Google Lens.

Como se puede apreciar en la Figura 1, podemos apuntar la cámara a un texto en otro idioma y pasados unos segundos, se mostrará la traducción por encima del mismo.

Algo realmente útil y que abre la puerta a perder los mie-

• E-mail de contacte: david.sanchezgon@autonoma.cat
• Menció realitzada: Computació
• Treball tutoritzat per: Coen Antens (departament)
• Curs 2021/22

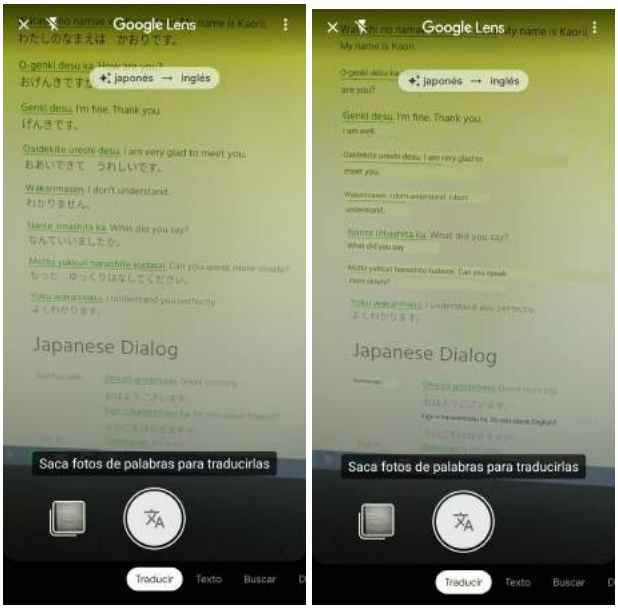

Fig. 1: Ejemplo de texto en japonés con su traducción.

dos sobre viajar a países donde no conocemos del todo el idioma y podemos perdernos, o no saber realmente lo que pedir en un restaurante, cuándo lo único necesario es apuntar a la carta del mismo para poder ver en tiempo real la traducción de la misma.

Por defecto se traducirá al idioma con el que se encuentre configurado el dispositivo, en este caso he escogido que se haga la traducción al inglés para hacer la comparación de la traducción original de la página web con la traducción que proporciona Google Lens.

Pero cuando se habla de Google Lens, no solo nos referimos a la traducción de textos, también existe la posibilidad de detectar ciertos monumentos o edificios históricos o conocidos y obtener más información sobre ellos.

Además de otras múltiples posibilidades, como encontrar el nombre de un objeto, flor, animal, etc. Abriendo las puertas a ampliar el conocimiento de las personas gracias a los pequeños ordenadores de los que disponemos en nuestros bolsillos.

Como se puede entender, Google Lens es una aplicación móvil que hace uso de una red de reconocimiento de texto de uso general, aunque también es posible este tipo de redes para obtener mejores resultados en algo más específico, como puede ser la detección de texto e objetos en una valla publicitaria[16].

2 Objetivos

El objetivo principal de este proyecto es el de indagar en los aspectos más importantes que componen Google Lens.

Para ello este documento se centrará principalmente en lo relacionado con el texto, y con ello en los tres aspectos que se consideran clave para el correcto funcionamiento de Google Lens cuando se le pasa una imagen, y en los que se dividirá principalmente el proyecto, estos son:

- **Detección**: Se deberá detectar correctamente la posición de los textos contenidos en la imagen.
- **Extracción**: Ser capaz de extraer el texto tal y como se muestra.
- **Traducción**: Traducir no solo el texto contenido en la imagen, sino que sea capaz de traducir correctamente texto que el usuario introduzca.

Para ello se han escogido tres datasets de imágenes especialmente pensados para pruebas de relacionadas con la detección y extracción de textos, además de un conjunto de imágenes reunidas por mí que simulan los entornos en los que será utilizado Google Lens, dataset del que se hablará más adelante en este documento.

TABLA 1: Planificación del proyecto

Tasca	Duració
Identificación e investigación sobre los diferentes tipos de algoritmos que pueden conformar Google Lens	2 semanas
Preparación de los tests para realizar las comparaciones	3 semanas
Realizar la comparación y análisis de los algoritmos	10 semanas
Elaboración de los informes de seguimiento	3 semanas
Elaboración del informe y presentación final	2 semanas
Elaboración del poster	2 semanas
Tiempo total	22 semanas

3 Planificación

Antes de comenzar con el proyecto se ha hecho una planificación sobre el tiempo estimado necesario para llevar a cabo todas las tareas necesarias. Esto se ha hecho teniendo en cuenta que hay un período de aproximadamente 22 semanas desde la reunión inicial con el tutor hasta la defensa del trabajo ante el tribunal. El desglose es el que se muestra en la Tabla 1.

A continuación se hace una breve explicación de cada apartado que se puede ver en la Tabla 1 previamente mencionada.

También se puede observar en la Figura 2 el diagrama de Gantt con una representación visual de las tareas a realizar en el espacio de tiempo designado para realizar el trabajo.

Explicando brevemente los pasos mostrados en la Tabla 1, se comienza investigando los diferentes algoritmos para cada sección del proyecto (detección, extracción y traducción de texto), además de identificar los más adecuados para cada una de ellas.

Una vez seleccionados los algoritmos/redes neuronales que se utilizarán, se procede a la creación y desarrollo de tests para cada sección, diferentes para cada una de ellas, pero iguales para los algoritmos escogidos en cada sección.

A continuación se ponen en marcha los tests previamente preparados en el apartado anterior, haciendo las correcciones necesarias en caso de surgir errores durante la ejecución de los mismos.

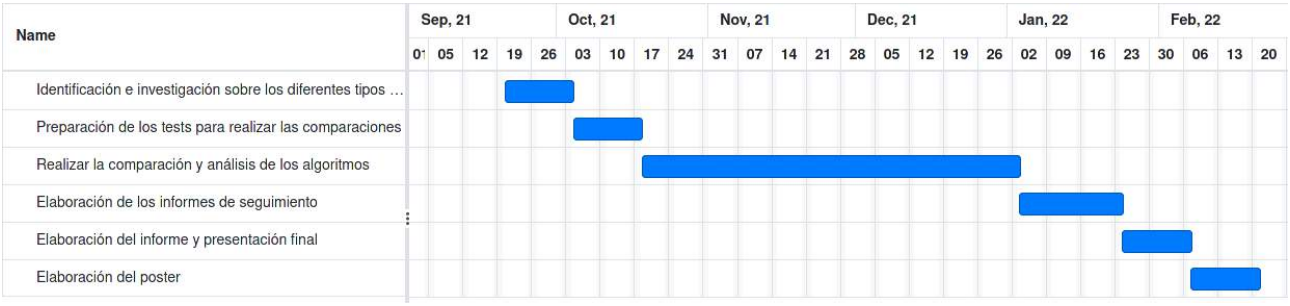

Fig. 2: Diagrama de Gantt con las tareas a realizar.

Una vez obtenidos todos los resultados, se efectúan las comparaciones entre los algoritmos y se extraen y comentan conclusiones sobre el rendimiento de los mismos.

4 Metodologia

El trabajo se desarrollará basándose en la metodología Agile, teniendo en cuenta que pueden surgir problemas durante la realización del trabajo, para los que se debe estar preparado para adaptarse en el tiempo total destinado al mismo.

Para ello se realizará una cantidad de trabajo y/o investigación durante la semana, que se pondrá en común en la reunión semanal con el tutor para mostrar aquello que se ha conseguido y comentar posibles dudas que hayan surgido durante el mismo período.

El orden que se puede ver en la tabla de planificación no es el orden en el que se harán las tareas, ya que elementos como la comparación y análisis de los algoritmos y la elaboración de los informes de seguimiento se han ido haciendo de forma alterna entre ambas, escribiendo en el informe sobre los resultados conseguidos en las diferentes pruebas.

Aunque ha habido semanas enfocadas en mayor parte a una de ambas, poniendo especial énfasis en la comparación y análisis, que supone la mayor carga de trabajo del proyecto.

5 Estado del arte

Todas las pruebas han sido creadas haciendo uso del lenguaje de programación Python, ya que ofrece mayor facilidad en el tratamiento de imágenes que lenguajes como C o C++, a pesar de no tener la velocidad característica de estos dos últimos.

Estas pruebas se han realizado en un cuaderno de Google Colab[15], plataforma que se puede utilizar para ejecutar notebooks de Python.

Google Colab es una plataforma que permite la creación, modificación y ejecución de notebooks de forma remota, haciendo uso de los recursos de las máquinas conectadas a los servidores de Google, lo que hace que máquinas con pocos recursos puedan ejecutar estos programas y recibir los resultados de los mismos en el propio navegador.

Como ya se ha comentado previamente, este proyecto se divide en tres apartados principales, en los que se entrará en detalle a continuación.

5.1. Detección de texto

En esta sección se ha decidido hacer uso de MSER y EAST Detector. Si bien es cierto que podría haberse utilizado en este apartado Keras y Tesseract, considero como mejor opción comparar dos algoritmos que solo detecten texto.

5.2. Extracción de texto

Para la extracción de los textos contenidos en las imágenes, las opciones más fiables y testadas son las que se basan en OCR[4] (*Optical Character Recognition*). Para ello existen multitud de opciones, como son Tesseract[5], OCRopus[6], Kraken[7], Keras OCR[8] y Easy OCR[9].

En concreto, se utilizarán Tesseract y Keras, al ser las dos redes con mayor documentación y soporte para multitud de idiomas, aunque realmente solo se utilizará para textos en inglés, español y catalán, soportados por la mayoría de redes centradas en el lenguaje. Además, que ambas tienen un ámbito de extracción de texto más general y no centrada tanto en documentos como puede ser OCROpus.

5.3. Traducción de texto

Una de las redes neuronales utilizadas para estas pruebas es la llamada GPT-3, desarrollada por OpenAI[12]. Se trata de una red capaz de generar texto que continúe la palabra o frase que reciba como entrada.

Su uso es gratuito e ilimitado, aunque se requiere crear una cuenta proporcionando diferentes datos personales, que la desarrolladora afirma es para controlar el acceso a la misma y no saturar sus servidores.

Aunque sea de uso general también puede usarse para la traducción de textos de manera sencilla, como una conversación entre dos o más idiomas. Un ejemplo de esto último sería introducir como entrada en GPT-3 ”English: hello Spanish:”, lo que en este caso devolvería ”hola”.

Otra de las redes empleadas es la llamada DeepL, la diferencia respecto a GPT-3 es que no es una red de uso general como es GPT-3, sino que está dedicada especialmente a la traducción de textos.

Para hacer uso de ella es necesaria crear una cuenta para controlar el acceso a la API al igual que en el caso de GPT-3. En esencia su uso también es gratuito, pero con una limitación de traducción de 500.000 caracteres.

6 DESARROLLO

Respecto a MSER, se trata de un algoritmo que, a pesar de no estar destinado a la detección de texto, sino a la detección de características, creo que sería posible adaptarlo para otorgarle esa habilidad, ya que se trata de un algoritmo que detecta rápidamente las características en las imágenes, pero es algo que se sale del ámbito del proyecto.

Por otra parte, EAST Detector se basa en el uso de una red neuronal pre-entrenada para la detección de texto, en el caso de estas pruebas, ha sido previamente entrenada en inglés para detectar texto en imágenes con diferentes situaciones y escenarios.

Cambiando a la extracción de texto, Tesseract está escrito en los lenguajes de programación C/C++ lo que a priori le otorga la posibilidad de obtener un mayor rendimiento en términos de velocidad a su contraparte Keras.

Para poder ser usado en Python, dispone de una capa llamada PyTesseract, que se encarga de la comunicación entre Python y Tesseract, permitiendo también escribir los resultados por pantalla y no únicamente escribirlos a un fichero de texto.

Por otro lado, Keras OCR es una capa que emplea Keras como su principal motor, esta una API escrita en Python que a su vez hace uso de la conocida plataforma de *Machine Learning* Tensorflow.

En el caso de la traducción de texto, ambas redes disponen de una API pública, cuyas conidiciones de uso se han explicado brevemente en el apartado anterior, por lo que no es necesario ejecutar nada con los recursos de tu máquina salvo la consulta a esta misma API con el texto a traducir para ver la traducción obtenida.

7 DATASETS

Para las pruebas realizadas se han hecho uso de cuatro datasets, tres de ellos siendo públicos y uno de ellos siendo un conjunto de imágenes que contienen texto en diferentes formas y ángulos reunidas por mi mismo.

En el caso de DDI-100[10], el dataset consiste en las páginas escaneadas de diferentes libros científicos rusos. En la Figura 3 se muestra un ejemplo de las imágenes contenidas dentro del mismo. El dataset completo tiene miles de imágenes diferentes, dónde se incluyen las cajas de texto de cada imagen, pero para las pruebas se utilizarán las imágenes originales sin ningún tipo de alteración o mejora de calidad.

El dataset MJSynth[11] cuenta con millones de imágenes, dónde cada una de ellas contiene una sola palabra que puede ser un recorte de periódico, una palabra escrita a mano o simplemente una palabra usando las fuentes que se usan en los ordenadores. No todas ellas se muestran rectas, pueden estar en diferentes ángulos y con transformaciones que pueden dificultar la detección de las mismas, tal como se puede apreciar en la Figura 4.

También he considerado interesante el hacer una recopilación de imágenes propia, simulando el que sería el entorno de uso estándar para una aplicación como Google Lens.

Para simular ese entorno se han realizado dos conjuntos de fotografías, una mitad interiores y la otra mitad exteriores. La primera mitad contiene imágenes de menús de restaurante, productos que se pueden encontrar en una tienda,

Fig. 3: Ejemplo de una imagen del dataset DDI-100.

Fig. 4: Ejemplo de una imagen del dataset MJSynth.

etc. Mientras que la segunda mitad contiene imágenes exteriores, como señales de tráfico que contengan letras, letreros de restaurantes, gasolineras y tiendas entre otros.

Por último, se hará uso del dataset llamado SVT[14], estas imágenes han sido extraídas a través de Street View de Google Maps. En general contiene letreros de diferentes establecimientos como restaurantes, gasolineras, supermercados, etc. Todas ellas tienen texto. En la Figura 6 se puede ver un ejemplo del mismo.

En cualquiera de las pruebas, solo se utilizarán un máximo de 50 imágenes para cada una de ellas, escogiendo las mismas imágenes de cada dataset para todas las pruebas, las cuales se usaran para hacer las comparaciones con los diferentes algoritmos de detección, extracción y traducción de texto.

8 RESULTADOS

Tal como se ha comentado previamente, en esta sección se mostrarán los resultados obtenidos por cada algoritmo en los diferentes datasets en las diferentes pruebas.

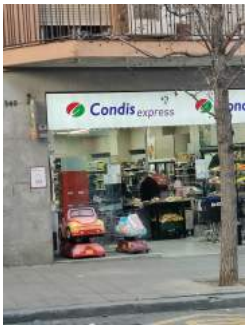

Fig. 5: Ejemplo de una imagen recopilada para el dataset propio.

Fig. 6: Ejemplo de una imagen recopilada para el dataset SVT.

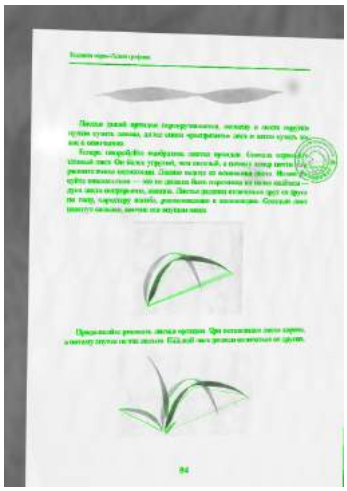

Fig. 7: Resultado obtenido usando MSER en el dataset DDI-100.

8.1. Detección de textos

Esta parte estará centrada exclusivamente a la calidad en la detección de los textos que puedan contener las imágenes. La calidad la marcará como de bien se encapsule el texto dentro de las cajas.

8.1.1. MSER

Aclarar que en estas pruebas no se ha hecho ninguna modificación a los algoritmos, pero se puede ver como detecta las letras contenidas en el texto, aunque no sea como palabras enteras.

Como se puede ver en la Figura 7 se detecta correctamente las áreas donde se encuentra el texto para el dataset DDI-100, pero tal y como se ha mencionado previamente, al ser un detector de características, también considera los bordes de los logos como texto, al ser estos fácilmente detectables por un detector de características como es MSER.

Para el dataset propio tenemos un buen ejemplo del funcionamiento de MSER, en la Figura 8 se puede apreciar como en la imagen la mayoría de regiones detectadas son las letras contenidas en la imagen, aunque también contenga regiones detectadas que no pertenecen a ningún texto.

Con SVT se puede observar en la Figura 9 un resultado similar al obtenido en la imagen del dataset propio, si bien encapsula decentemente las letras, también encapsula otras características de la imagen, cosa esperada teniendo en cuenta la naturaleza del algoritmo.

Fig. 8: Resultado obtenido usando MSER en el dataset propio.

De nuevo y como se puede ver en la Figura 10 para el dataset MJSynth, se puede ver como tenemos una imagen en la que se encapsulan a la perfección todas las letras de la palabra, en este caso “NORMA”, y otra imagen en la que se ve que acierta respecto a la región en la que se encuentra el texto, pero no captura bien las letras de forma independiente.

A pesar de que el rendimiento no es tan importante, considero que es interesante mencionar que la velocidad de ejecución del algoritmo depende de la resolución de la imagen, por lo que necesita cerca de un segundo por imagen en los datasets propio y DDI-100, y una décima de segundo para cada imagen del dataset MJSynth.

8.1.2. EAST Detector

Para el dataset DDI-100 es posible ver que el algoritmo funciona bastante bien, a pesar de que es posible ver falsos positivos tanto en los logos como en las imágenes mostradas en las páginas como se puede ver en la Figura 11, por lo general encapsula correctamente la mayoría de palabras dentro de la página.

En el caso del dataset propio se han obtenido resultados que pueden considerarse bastante buenos como el de la Figura 12, sin embargo, en el caso de imágenes en las cuales los textos no están en una posición recta, necesita dibujar una o más cajas para encapsularlo o no posiciona del todo correcto la caja de texto, pero acercándose mucho, y, aunque no lo haga de forma completa, se podría considerar como un buen resultado. Esto último se puede ver con claridad en la Figura 13.

Por desgracia no se han obtenido muchos resultados haciendo uso de este algoritmo para el dataset MJSynth, algo

Fig. 9: Resultado obtenido usando MSER en el dataset SVT.

Fig. 10: Resultado obtenido usando MSER en el dataset MJSynth.

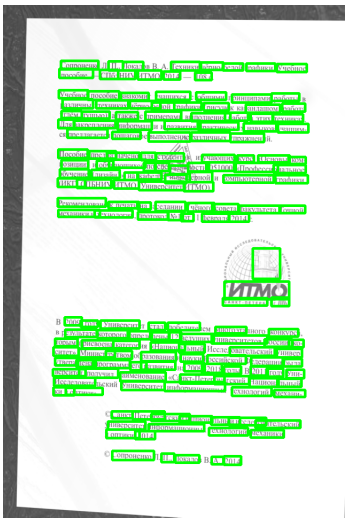

Fig. 11: Resultado obtenido usando EAST Detector en el dataset DDI-100.

que quizás pueda ser achacado a la muy baja resolución de las imágenes. Tal y como se puede observar en la Figura 14, las detecciones son parciales y no muy bien posicionadas.

Respecto al rendimiento, con este algoritmo la velocidad era prácticamente idéntica para cada dataset, sin importar la resolución.

8.2. Extracción de texto

En este apartado se hará una comparación entre las redes naturales para extracción de texto de las imágenes. Estas pruebas se centrarán en los datasets MJSynth y el dataset propio.

Para ello crearé archivos de texto que servirán como ground truth para cada imagen. En el caso del dataset propio, se hará a mano, con una palabra por cada línea del fichero, sin importar el orden de lectura, ya que se efectuará la comprobación de las palabras que han sido correctamente extraídas, sin tener en cuenta la frase a la que puedan pertenecer.

En el caso de MJSynth los ficheros de las imágenes contienen la palabra contenida en las mismas en el nombre del fichero. Tal como se explica en la sección 7, cada imagen solo contiene una palabra, por lo que es posible extraer la palabra del nombre del fichero.

Se han seleccionado un total de 50 imágenes al azar, las cuales serán siempre las mismas para todas las pruebas.

Una vez generados los ficheros para el ground truth, se realizará una comparación palabra por palabra, donde al final se obtendrá un porcentaje que indicará el rendimiento del algoritmo con ese dataset en concreto, que teniendo en cuenta también el número de imágenes de las que los algoritmos son capaces de extraer texto, este será calculado de la siguiente manera:

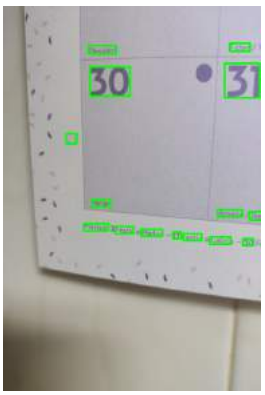

Fig. 12: Resultado obtenido usando EAST Detector en el dataset propio.

Fig. 13: Resultado obtenido usando EAST Detector en el dataset SVT.

$$\frac{TExtraidos}{ITotal} * \frac{PCorrectas}{PTotales} * 100$$

Dónde *TExtraídas* es el número de imágenes de las que ha sido capaz el algoritmo de extraer texto, y *ITotal* el total de imágenes utilizadas para las pruebas, en este caso y tal como se ha comentado previamente, será 50, *PCorrectas* son aquellas palabras que coinciden exactamente con las de su correspondiente groundtruth, *PTotales* el total de palabras contenidas en los ground truth para las imágenes en las cuales los algoritmos han conseguido extraer texto.

8.2.1. MJSynth

Aunque lo más importante es la calidad de la extracción de texto, también es interesante comentar la eficiencia de ambos algoritmos. En el caso de Tesseract, de las 50 imágenes Tesseract ha conseguido extraer texto en 40 imágenes, mientras que Keras ha conseguido extraerlo en 48.

En el apartado de rendimiento, Keras obtiene también mejores resultados que Tesseract, extrayendo un total de 42 palabras correctas de 48 posibles, contra las 17 de 40 de Tesseract.

Por lo que el porcentaje de rendimiento total de ambos, teniendo en cuenta lo explicado previamente, es de 35 % para Tesseract y 84 % para Keras.

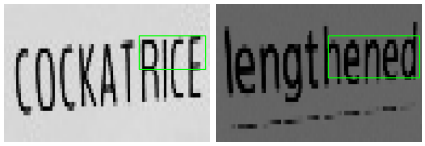

Fig. 14: Resultado obtenido usando EAST Detector en el dataset MJSynth.

TABLA 2: CONJUNTO DE RESULTADOS OBTENIDOS CON TESSERACT PARA EL DATASET MJSYNTH

Esperado	Obtenido
classmates	cossmates
greater	greatel
nuclear	nuclear,
maximilian	maximilian
reformulated	reformulated
expend	expend
reweaves	reweaves
biophysics	s, nvaete
locke	lucke
euphemistically	cuphemistically

TABLA 3: CONJUNTO DE RESULTADOS OBTENIDOS CON KERAS PARA EL DATASET MJSYNTH

Esperado	Obtenido
classmates	classmates
greater	greater
nuclear	nuclear
maximilian	maximilian
reformulated	reformulated
expend	expend
reweaves	ves, rewea
biophysics	biophysics
locke	locke
euphemistically	euphemistically

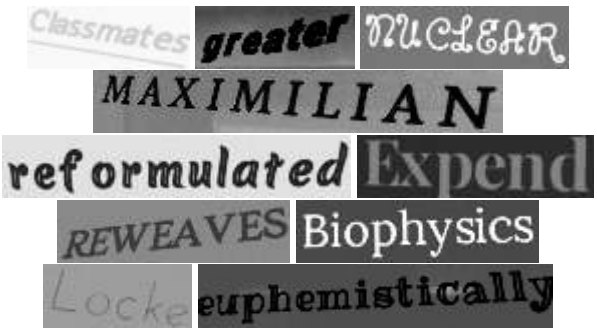

Fig. 15: Imágenes utilizadas para las Tablas 2 y 3

Fig. 16: Imágenes utilizadas para las Tablas 4 y 5

En las Tablas 2 y 3 se pueden contemplar las palabras extraídas por Tesseract y Keras para el dataset MJSynth respectivamente.

En ellas se puede ver como, si bien es cierto que Keras ha extraído más palabras correctas, Tesseract en su mayor parte y como se ve en la Tabla 2 solo ha fallado en un carácter de la palabra en varias ocasiones, lo que obviamente hace contar la palabra extraída como incorrecta.

Pero teniendo en cuenta las imágenes de las cuales se ha extraído, es un error comprensible por parte de ambos, ya que palabras como "classmates" y "euphemistically" en la Figura 15 es fácil confundir algún carácter por la iluminación o por ser estos muy parecidos entre ellos.

Teniendo esto último en cuenta, es posible decir que Keras todavía sale vencedor, pero no con tanta diferencia como hacía prever las estadísticas mostradas anteriormente en este mismo apartado.

8.2.2. SVT

Con este dataset se han obtenido unos resultados realmente malos, con solo 4 imágenes dónde se haya extraído texto, y solo una de ellas ha sido correcta. En la Tabla 4 se aprecia lo que se esperaba extraer y los resultados obtenidos. Mientras que en la Figura 16 las imágenes de las cuales se ha extraído texto.

Para este dataset y los siguientes solamente se mostrarán los resultados de cinco imágenes, ya que estas tienen un tamaño bastante superior a las del dataset MJSynth, por lo que considero que ocuparían demasiado espacio en el documento y no se podría apreciar el texto en caso de hacerlas aún más pequeñas.

De nuevo se obtiene un mejor resultado con Keras en comparación con Tesseract. En este caso ha extraído correctamente 55 de un total de 216 palabras, es decir, un 25 % de ellas, que si bien es cierto no es un resultado brillante, es mucho mejor que el obtenido por Tesseract. Y en el apartado caracteres ha detectado 401 de un total 1114, lo que significa un 36 % de acierto sobre las palabras que comparten la misma longitud entre palabra esperada y detectada.

En la Tabla 5 se pueden ver ejemplos de los resultados obtenidos, mientras que, como ya se ha mencionado anteriormente, se pueden observar las imágenes utilizadas en la Figura 16.

8.2.3. Dataset propio

Se vuelve a repetir el hecho de obtener peores resultados con Tesseract sobre Keras, en este caso solo se ha conseguido extraer correctamente 1 palabra y solo se ha conseguido extraer texto de dos de las imágenes.

Se muestran los resultados obtenidos para esta prueba en la Tabla 6 y en la Figura 17 las imágenes de las cuales se han obtenido.

Para Keras se tienen de nuevo unos resultados más que

TABLA 4: CONJUNTO DE RESULTADOS OBTENIDOS CON TESSERACT PARA EL DATASET SVT

Esperado	Obtenido
avante, gardens, florist, school, of, floral, design	1, mass
paul, laurence, dunbar, community, high, school, 1400, orleans, street	eee, eer, al, ej
chase, brexton, heath, services,, inc.	of, chase-brexton
thairish, thai, cuisine	ath
red, square, sops, belvedere	itt, mo:

TABLA 5: CONJUNTO DE RESULTADOS OBTENIDOS CON KERAS PARA EL DATASET SVT

Esperado	Obtenido
avante, gardens, florist, school, of, floral, design	gardens, avante, florist, ullee, school, desig, loral
paul, laurence, dunbar, community, high, school, 1400, orleans, street	dunbar, laurence, paul, school, high, community, 1400, orleans, street
chase, brexton, heath, services,, inc	chasebrexton, health, services, inco
thairish, thai, cuisine	oaiih, tnai, cuisine
red, square, sops, belvedere	belvedere, red, suuare, sops, 025

TABLA 6: CONJUNTO DE RESULTADOS OBTENIDOS CON TESSERACT PARA EL DATASET PROPIO

Esperado	Obtenido
agua, mineral, natural, viladrau, l'aigua, que, ens, mou	
explorer, mont, blanc	
febrer, 2022, dimecres, dijous, divendres, 3, 4	dimecres
mapfre, mapfre, ograma, el, teu, futur, porta, la teva, jubilació	wo
uso, exclusivo, de, correos, ús, exclusiu, de, correos	

TABLA 7: CONJUNTO DE RESULTADOS OBTENIDOS CON KERAS PARA EL DATASET PROPIO

Esperado	Obtenido
agua, mineral, natural, viladrau, l'aigua, que, ens, mou	mineral, agua, natural, viladrau, ens, lnigun
explorer, mont, blanc	explorer, mont, blanc
febrer, 2022, dimecres, dijous, divendres, 3, 4	felorer, 2022, dimecres, dijous, divendres
mapfre, mapfre, ograma, el, teu, futur, porta, la teva, jubilació	mapfre, eufutua, ograma, unuacio, lnuld, mapfre
uso, exclusivo, de, correos, ús, exclusiu, de, correos	correos, exclusivo, correos, exclusiu, uso

aceptables, sobrepasando en este caso el 50 % de palabras acertadas, en concreto el 52 %, consiguiendo con esto doblar el porcentaje obtenido por Keras en el dataset SVT. En el apartado de carácter es el resultado es inferior, siendo un 26 % de caracteres acertados, pero aun así considero que estos resultados son mejores que los obtenidos con el dataset SVT debido al alto porcentaje de palabras completas correctas conseguido.

En la Tabla 7 se encuentran resultados obtenidos para Keras y en la Figura 17 las imágenes con las que se han conseguido los mismos.

8.2.4. Conclusiones

Como se ha comentado en los dos apartados anteriores, se ha podido observar que para los datasets utilizados para realizar estas pruebas (MJSynth, SVT y el dataset propio), se han obtenido mejores resultados con Keras.

Es cierto que Tesseract ha sido de media de 3 a 10 veces más rápido por imagen, y que esa velocidad lo haría más probable a ser escogido para su implementación en una app como es Google Lens, en la que el rendimiento de la misma sería primordial. Pero no se puede pasar por alto el hecho del pésimo rendimiento conseguido con ambos datasets, en los cuales apenas ha conseguido extraer correctamente el texto de las imágenes.

Por otro lado, se podría aceptar el tiempo de procesamiento de Keras alegando los resultados bastante aceptables que se han obtenido con ambos datasets, en especial con MJSynth. Teniendo en cuenta que realmente 5 segundos por imagen es bastante elevado en comparación con las 2 décimas de segundo de media de Tesseract, se trata de un tiempo relativamente reducido para los buenos resultados obtenidos.

Cabe remarcar que ambos algoritmos han extraído más palabras de las que realmente existen en la imagen, esto sucede porque en ocasiones detectan una palabra como múltiples de ella, ya que no detecta correctamente los caracteres contenidos en la misma. No se ha mencionado previamente porque lo realmente importante era encontrar el número de palabras correctas que eran capaces de detectar ambos algoritmos.

8.3. Traducción de textos

En este caso se realizará unas pruebas más bien subjetivas, ya que hay casos en los que las imágenes contienen frases en lugar de palabras sueltas, por lo que las traduccio-

Fig. 17: Imágenes utilizadas para las Tablas 6 y 7

nes pueden ser diferentes e igual de correctas, por lo que no será posible automatizarlo de la misma forma que en las anteriores pruebas.

Para este apartado se comentarán las diferentes traducciones obtenidas por los diferentes algoritmos utilizados, dando como buenas aquellas que tengan coherencia y sean subjetivamente buenas traducciones de los textos originales.

Las pruebas se harán haciendo uso de las palabras que se contienen las imágenes, es decir, se usarán las palabras y frases contenidas en los ground truths generados para cada dataset, además de contar con los resultados derivados de la traducción de frases hechas escogidas al azar.

8.3.1. GPT-3

En la Tabla 8 se puede comprobar una muestra de 20 traducciones realizadas por GPT-3. En esta misma se encuentran mezcladas palabras de los diferentes ground truths de cada imagen y frases hechas, tal como se ha comentado en la sección anterior.

8.3.2. DeepL

Al igual que en el apartado anterior, se le han pasado las mismas palabras y frases que a GPT-3, pudiendo observar los resultados en la Tabla 9.

TABLA 8: CONJUNTO DE RESULTADOS OBTENIDOS CON GPT-3 DE OPENAI

Entrada	**Traducción**
playground	parque de juegos
ostrich	camafeo
going to the mall	ir a la tienda
breaking the glass	romper el cristal
read	leer
drink water	beber agua
kevin likes to listen to rock	kevin le gusta escuchar rock
ice	hielo
spaceship	nave espacial
destroy	destruir
trigger	disparador
lantern	lámpara
i want to read this book	quiero leer este libro
travel	viaje
traveling to germany	viajar a Alemania
writing a novel	escribir una novela
making a movie	hacer una película
developer	desarrollador
dark	oscuro
smartphone	smartphone

TABLA 9: CONJUNTO DE RESULTADOS OBTENIDOS CON DEEPL

Entrada	**Traducción**
playground	parque infantil
ostrich	ostras
going to the mall	ir al centro comercial
breaking the glass	rompiendo el cristal
read	leer
drink water	beber agua
kevin likes to listen to rock	a kevin le gusta escuchar rock
ice	hielo
spaceship	nave espacial
destroy	destruir
trigger	desencadenar
lantern	lantern
i want to read this book	quiero leer este libro
travel	viajar
traveling to germany	viajar a alemania
writing a novel	escribir una novela
making a movie	hacer una película
developer	desarrollador
dark	oscuro
smartphone	smartphone

8.3.3. Conclusiones

Como se ha podido ver en ambas tablas, nos encontramos con resultados muy similares entre ambas redes neuronales.

Hay casos en los que las traducciones no son las mismas, como en el caso de ”trigger”, ya que se trata de una palabra con múltiples traducciones según el contexto en el que sea usado, por lo que en este apartado ambas han obtenido buenos resultados.

En una ocasión DeepL ha conseguido una mejor traducción, y esa es en la frase ”going to the mall”. GPT-3 en este caso traducido ”mall” como ”tienda”, cuando la traducción más correcta sería ”centro comercial”, tal y como se ha obtenido con DeepL.

Lo que los resultados para ambas redes parecen demostrar es que ambas tienen dificultades a la hora de conjugar correctamente los verbos de las frases. Ambas han cogido el infinitivo en lugar del gerundio como correspondería en varias de las frases y palabras que se les ha pasado, tres de ellos se pueden ver en ambas tablas: ”traveling to Germany”, ”writing a novel” y ”making a movie”.

Por lo general y a pesar de estos pequeños fallos, se puede concluir que ambas realizan un trabajo excelente a la hora de traducir textos de diferente longitud.

AGRADECIMIENTOS

Me gustaría dar las gracias en primer lugar a mi tutor, Coen Antens, por el soporte y consejos para mejorar la calidad del proyecto.

También me gustaría agradecer a mi familia y amigos por ser el apoyo moral que a veces ha sido necesario a lo largo de estos años.

REFERENCIAS

[1] Google Lens, https://lens.google/

[2] “Detecting Text in Natural Image with Connectionist Text Proposal Network”, Zhi Tian, Weilin Huang, Tong He, Pan He, Yu Qiao, https://arxiv.org/abs/1609.03605

[3] “EAST: An Efficient and Accurate Scene Text Detector”, Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang - 10.1109/CVPR.2017.283

[4] OCR, https://anyline.com/news/what-is-ocr

[5] “Document Segmentation and Language Translation Using Tesseract-OCR”, Sahil Thakare, Ajay Kamble, Vishal Thengne, U.R. Kamble - 10.1109/ICIINFS.2018.8721372

[6] OCRopus, https://github.com/ocropus

[7] Kraken, http://kraken.re/

[8] Keras OCR, https://keras.io/about/

[9] Easy OCR, https://github.com/JaidedAI/EasyOCR

[10] DDI-100 Dataset, https://github.com/machine-intelligence-laboratory/DDI-100/...

[11] MJSynth Dataset, https://tinyurl.com/MJSynth-VGG-Oxford

[12] OpenAI, https://openai.com/

[13] Dataset Propio, https://drive.google.com/file/d/1bpDZdtv...

[14] SVT Dataset, http://vision.ucsd.edu/˜kai/grocr/

[15] Notebook en Google Colab con el código utilizado para realizar las pruebas, https://colab.research.google.com/drive/13tW3ZpnC42...

[16] “Signboard Detection and Text Recognition Using Artificial Neural Networks”, Muhammad A. Panhwar, Kamran A. Memon, Adeel Abro, Deng Zhongliang, Sijjad A. Khuhro, Saleemullah Memon - 10.1109/ICEIEC.2019.8784625