GRADO EN INGENIERÍA EN TECNOLOGÍAS EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

HERRAMIENTA DE EXTRACCIÓN AUTOMÁTICA DE DATOS DE PÓLIZAS DE SEGUROS

Autor: Pablo García Bolívar

Director: Miguel Ángel Durán Olivencia

Madrid

Agosto 2022

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

‘Herramienta de extracción automática de datos de pólizas de seguros’

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2021/22 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Pablo García Bolívar Fecha: 30/08/2022

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Miguel Ángel Durán Olivencia Fecha: ……/ ……/ ……

GRADO EN INGENIERÍA EN TECNOLOGÍAS EN TECNOLOGÍAS INDUSTRIALES

TRABAJO FIN DE GRADO

HERRAMIENTA DE EXTRACCIÓN AUTOMÁTICA DE DATOS DE PÓLIZAS DE SEGUROS

Autor: Pablo García Bolívar

Director: Miguel Ángel Durán Olivencia

Madrid

Agosto 2022

HERRAMIENTA DE EXTRACCIÓN AUTOMÁTICA DE DATOS DE PÓLIZAS DE SEGUROS

Autor: García Bolívar, Pablo.
Director: Durán Olivencia, Miguel Ángel.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

Empresas de todo el mundo generan diariamente enormes cantidades de documentos que ralentizan el trabajo de sus empleados y cuya información generalmente queda archivada para siempre. Analizar dichos documentos con herramientas de *Document AI* y extraer su información de manera automatizada ayuda a mejorar el rendimiento empresarial. Sin embargo, ninguna de las soluciones existentes está centrada en pólizas de seguros, o requieren de cientos de pólizas anotadas manualmente para su correcto funcionamiento. Por ello, en este Trabajo de Fin de Grado se presenta una nueva herramienta de extracción automática de datos de pólizas de seguros, construida a partir de modelos de aprendizaje automático, que permite al usuario analizar sus pólizas sin necesidad disponer documentos anotados.

Palabras clave: *Document AI*, Póliza de seguro, Herramienta de extracción de datos, Inteligencia Artificial

1. Introducción

La inmensa cantidad de documentos generados, enviados y recibidos diariamente por las empresas tiene un impacto negativo en su rendimiento. Por un lado, los trabajadores deben emplear enormes cantidades de tiempo buscando información en documentos para satisfacer las necesidades de los clientes. En concreto, según un estudio realizado por ABBYY, en Reino Unido un 27% de los empleados perdían un día laboral entero por semana buscando dicha información. Por otro lado, de no ser extraída, la información contenida en dichos documentos, muchas veces de gran importancia para las empresas, quedará archivada para siempre. Por ello, la extracción de datos de documentos juega un rol muy importante en la mejora del rendimiento de las empresas.

La extracción manual de dicha información resulta ineficiente y poco escalable. Sin embargo, el *Document AI*, al emplear modelos de Inteligencia Artificial, ofrece una solución mucho más precisa y escalable, permitiendo a las empresas analizar documentos de manera automatizada. Los principales modelos utilizados en este ámbito están basados en la arquitectura de los *Transformers*. Generalmente, estos modelos son pre-entrenados con aprendizaje autosupervisado en tareas generales, y más tarde, afinados en tareas concretas con pequeños datasets de documentos anotados.

El creciente número de empresas que demandan los servicios de *Document AI* hace ver que se trata de un sector al alza. Empresas como Amazon, Google o Microsoft ofrecen sus servicios de extracción de datos en forma de plataformas de pago alojadas en la nube. Sin embargo, ninguna de estas soluciones está centrada en pólizas de seguros, o requieren de cientos de pólizas anotadas manualmente para su correcto funcionamiento.

2. Definición del proyecto

En este TFG se desarrolla una herramienta de extracción de datos de pólizas de seguros, construida a partir de modelos de aprendizaje automático, que solventa las carencias de las soluciones existentes. Esta herramienta debe ser capaz de extraer: (1) todo el texto presente en el documento, (2) todos los pares de datos relacionados y (3) elementos clave definidos por el usuario; todo ello sin necesitar entrenamiento por parte del usuario. Esta herramienta está escrita en Python y ejecutada en un cuaderno de Google Colab.

Los modelos de Inteligencia Artificial escogidos para el desarrollo de la herramienta comparten una serie características comunes. En primer lugar, son modelos de código abierto. En segundo lugar, deben poder ser descargados listos para su uso, lo que implica que deben de estar ya pre-entrenados y afinados. En tercer lugar, deben ofrecer buenos resultados en sus respectivas tareas, de no ser el estado del arte. Cumpliendo con estas características, el modelo LayoutLMv2 es el pilar central de la herramienta, y se utiliza con el objetivo de etiquetar cada *token* encontrado en el texto. Este modelo, desarrollado por Microsoft basado en la arquitectura BERT, incorpora información sobre el texto encontrado en el documento, su posición 2-D e información visual del mismo. A parte, también se utiliza el modelo PEGASUS de Google afinado en un *dataset* de comunicados y reclamaciones de litigios de la SEC. Este modelo tiene el objetivo de resumir párrafos de texto encontrados en el documento, con el fin de presentar la información lo más organizada y resumida posible. Además, se utiliza el modelo all-MiniLM-L6-v2 para hallar la similitud entre dos oraciones. Esto ayuda a extraer elementos clave del documento. Por último, se va a emplear la herramienta OCR Pytesseract, para extraer el texto del documento.

3. Descripción de la herramienta desarrollada

El proceso de extracción de datos desarrollado en este TFG comienza por importar el documento que se desea analizar. Este documento debe contener una sola página y estar en formato imagen. En la Figura 1 se muestra la arquitectura de la herramienta desarrollada y, por tanto, el proceso de análisis que seguirán los documentos introducidos.

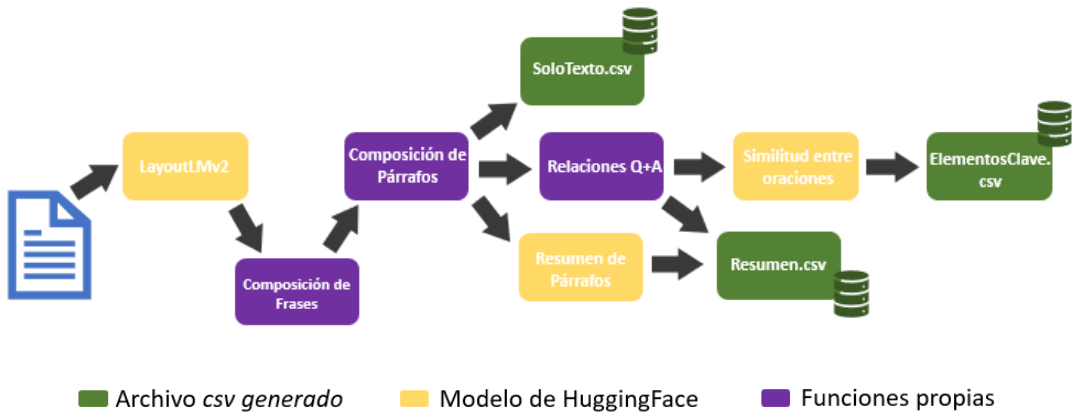

Figura 1. Arquitectura de la herramienta desarrollada.

El documento introducido es procesado en primer lugar por LayoutLMv2, que etiquetará cada *token* con una de las etiquetas {'Other', 'Header-Beginning', 'Header-Intermediate', 'Question-Beginning', 'Question-Intermediate', 'Answer-Beginning',

'Answer-Intermediate'}, como puede verse en la Figura 2. Estas etiquetas aportan información muy valiosa sobre el tipo de cada *token* y las posibles relaciones entre ellos. Sin embargo, LayoutLMv2 no establece dichas relaciones y etiqueta palabras de manera individual, pero no frases. Por ello, el siguiente paso es formar frases a partir de las palabras etiquetadas.

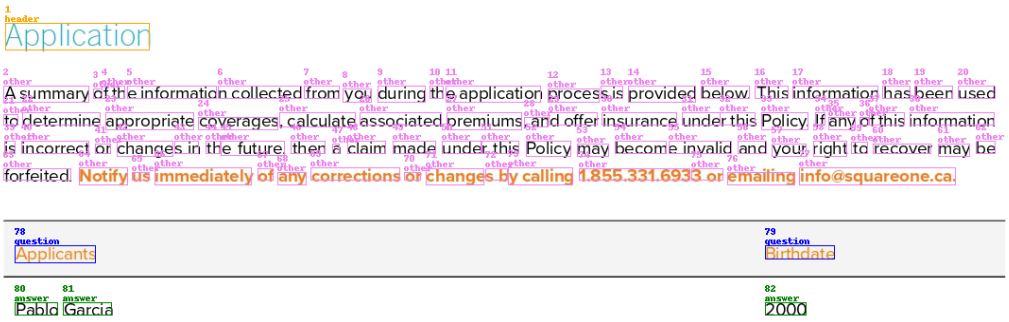

Figura 2. Ejemplo de etiquetado de una documento con LayoutLMv2.

Con este objetivo en mente, se ha creado una función que determina si una palabra y la de su derecha pertenecen a la misma frase o no. Dos palabras pertenecerán a la misma frase cuando estén una al lado de la otra y sean del mismo tipo {'Other', 'Header', 'Question', 'Answer'}. Una vez formadas las frases, se forman a continuación párrafos siguiendo un proceso muy similar. Los párrafos estarán compuestos por frases del mismo tipo, que estén una encima de la otra. Con los párrafos ya formados, en primer lugar, se procede a extraer el texto de cada uno de ellos con la herramienta OCR Pytesseract y a guardarlo dicho texto en el archivo *SoloTexto.csv*. Este archivo contendrá todo el texto encontrado en el documento, cumpliéndose así el primer objetivo propuesto.

Por otro lado, los párrafos de tipo 'Other' son procesados por el modelo legal-PEGASUS, que los resume sin que exista pérdida de información relevante. Paralelamente, los párrafos, esta vez todos ellos, y no solo los de tipo 'Other', son procesados por una función que encuentra relaciones formando pares con el formato 'Pregunta + Respuestas'. Esta función busca estructuras determinadas partiendo de la hipótesis de que la respuesta asociada a una pregunta se encontrará a su derecha o hacia abajo. Las relaciones encontradas por esta función junto con los párrafos resumidos de tipo 'Other' son guardados en un nuevo archivo llamado *Resumen.csv*, que, como su propio nombre indica, sirve a modo de resumen de toda la información contenida en el documento.

Por último, la herramienta analiza la similitud entre las preguntas de los pares 'Pregunta + Respuestas' y una serie de diccionarios con elementos que se pretenden extraer del documento. La herramienta incorpora de manera predeterminada tres diccionarios: un diccionario genérico a todas las pólizas de seguros, un diccionario específico para seguros del automóvil y un diccionario específico para seguros del hogar. Sin embargo, la creación de nuevos diccionarios personalizados es posible. Para cada elemento encontrado en cada uno de los diccionarios se selecciona la pregunta con la que tenga mayor similitud. De esta manera, la herramienta es capaz de encontrar los elementos del documento que más se parecen a cada elemento que se desea extraer de él. Todos estos elementos extraídos son guardados en un último archivo llamado *ElementosClave.csv*.

4. Resultados

En este análisis de resultados se van a analizar, en primer lugar, los resultados obtenidos al procesar dos pólizas de seguros distintas con la herramienta desarrollada en este TFG. Posteriormente, se van a comparar dichos resultados con los obtenidos al analizar las mismas pólizas con la herramienta Document AI de Google.

En la primera tarea objetivo, la extracción de todo el texto presente en el documento, la herramienta desarrollada en este TFG es capaz de extraer correctamente el 89% de las frases en una póliza y el 60% de la otra. Contando cada póliza con 82 y 77 frases a extraer, respectivamente. A la hora de formar relaciones entre elementos del documento, la herramienta extrajo 27 relaciones de la primera póliza y 16 de la segunda. En la Tabla 1 se muestran parte de las relaciones extraídas de la primera póliza.

PREGUNTA	RESPUESTA 1	RESPUESTA 2
Pollcy number	12345~67-89	
Effective	(01/04/2019 12:07 AM.	
Expiration	06/01/2019 12:01AM	
Named insured(s)	Jack Smith	Jane Smith
Underwritten by	Farmers Insurance	6301 Owensmouth Ave.
Policy premium + fees:	$798.48	
Veh. # year/make/model/vin	2019 Porsche Macan 4D 4wD-	
Bodily injury	$250K each person/$500K each incident	
Property damage	$100K each incident	Included
Medical coverage	$5,000 each person	$66.50
Deductible	3500	ACV
Collision	$1,000	ACY

Tabla 1. Relaciones extraídas por la herramienta.

Por último, en la tarea de extracción de elementos clave, la herramienta extrajo 12 elementos de los 14 elementos objetivo de la primera póliza y 2 de 9 en la segunda póliza. En la Tabla 2 se pueden ver algunos de los elementos extraídos de la primera póliza de seguro.

DICCIONARIO GENÉRICO - SEGURO DEL AUTOMÓVIL	
Policy number	12345-67-89
Name insured	Jane Smith
Effective date	01/01/2019 0:01
Premium	3797.6
Address	1234 Main St
Limit	$250K each person/$800K each incident

Tabla 2. Elementos clave extraídas por la herramienta.

Por su parte, Document AI de Google, utilizando la plantilla de documento genérico, es capaz de extraer a la perfección el texto mediante OCR, sin cometer errores. Sin embargo, a la hora formar relaciones, esta herramienta solo formó 18 relaciones en la primera póliza y 2 en la segunda. Por último, la plantilla genérica utilizada para el análisis de estas pólizas no ofrece la posibilidad de extraer elementos clave, y la creación de una platilla personalizada que sí lo permita requeriría de al menos 200 pólizas con los elementos clave anotados manualmente.

Se puede concluir que la herramienta de Google incorpora una gran herramienta de OCR, pero tiene problemas a la hora de formar relaciones y no ofrece la posibilidad de extraer elementos clave. En cambio, la herramienta desarrollada en este TFG establece de manera satisfactoria relaciones entre elementos del documento y es capaz de extraer gran cantidad de los elementos clave objetivo; sin embargo, tiene problemas a la hora de extraer el texto con precisión. Cabe remarcar que esta falta de precisión a la hora de extraer texto empeora también los resultados obtenidos en el resto de las tareas.

5. Conclusiones

Como conclusiones generales, se ha podido probar la eficacia de los modelos basados en *Transformers*, entrenados con el método de pre-entrenamiento más afinado. Estos modelos destacan por su capacidad de entendimiento del lenguaje humano y por la versatilidad de sus aplicaciones. También se ha podido comprobar cómo introducir elementos visuales y de posición aumenta el rendimiento de los modelos en tareas del *Document AI*. Además, se ha observado de manera práctica que una herramienta precisa de OCR es un elemento clave para este tipo de herramientas, y se le debe dar prioridad.

Como trabajos futuros para mejorar la herramienta presentada en este TFG se aconseja, en primer lugar, sustituir LayoutLMv2 por el recién publicado LayoutLMv3, ya que este último ofrece mejores resultados al etiquetar *tokens*. En segundo lugar, puede resultar interesante la adaptación de la herramienta a otros idiomas, especialmente a español. Esto se puede conseguir incorporando el modelo LayoutXLM, la versión plurilingüe de la misma familia. En tercer lugar, la mejora en la implementación y uso de Pytesseract en esta herramienta sería muy beneficiosa y mejoraría significativamente los resultados obtenidos. Por último, sería conveniente el desarrollo de una función que permita analizar documentos de diversas páginas, así como analizar documentos por lotes.

6. Referencias

Bao, H., Dong, L., & Wei, F. (2021). *BEiT: BERT Pre-Training of Image Transformers.* arXiv:2106.08254.

Cui, L., Xu, Y., Lv, T., & Wei, F. (2021). *Document AI: Benchmarks, Models and Applications.* arXiv:2111.08609v1.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transfromers for Languaje Understanding.* arXiv:1810.04805v2.

Huang, Y., Lv, T., Cui, L., Lu, Y., & Wei, F. (2022). *LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking.* arXiv:2204.08387.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). *Attention Is All You Need.* arXiv:1706.03762.

Walker, A. (2021). Document Searches Can Waste 100s of Hours. Obtenido de https://insurance-edge.net/2021/11/10/document-searches-can-waste-100s-of-hours/

Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., & Zhou, M. (2020). *MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers.* arXiv:2002.10957v2.

Xu, Y., Xu, Y., Lv, T., Cui, L., Wei, F., Wang, G., . . . Zhou, L. (2021). *LayoutLMv2: Multi-modal Pre-training for Rich Document Understanding.* arXiv:2012.14740v3.

Zhang, J., Zhao, Y., Saleh, M., & Liu, P. J. (2020). *PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization.* arXiv:1912.08777v3.

TOOL FOR AUTOMATIC EXTRACITION OF DATA FROM INSURANCE POLICIES

Author: García Bolívar, Pablo.
Supervisor: Durán Olivencia, Miguel Ángel.
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

Every day, companies around the world generate huge amounts of documents that slow down the work of their employees and whose information is often archived forever. Analysing these documents with Document AI tools and extracting their information in an automated way helps to improve business performance. However, none of the existing solutions focuses on insurance policies, or require hundreds of manually annotated policies to work properly. Therefore, this thesis presents a new tool for automatic extraction of data from insurance policies, built with machine learning models that allows users to analyse their policies without the need of providing annotated documents.

Keywords: Document AI, Insurance Policy, Data Mining Tool, Artificial Intelligence

1. Introduction

The immense number of documents generated, sent and received on a daily basis by companies has a negative impact on their performance. On the one hand, workers must spend enormous amounts of time searching for information in documents to meet customer needs. Specifically, according to a study conducted by ABBYY, 27% of employees in the United Kingdom lost an entire working day per week searching for such information. On the other hand, if it is not extracted, the information contained in these documents, often of great importance to companies, will remain archived forever. Therefore, the extraction of data from documents plays a very important role in improving the performance of companies.

Manual extraction of such information is inefficient and not very scalable. However, Document AI, by employing Artificial Intelligence models, offers a much more accurate and scalable solution, allowing companies to analyze documents in an automated way. The main models used in this field are based on the Transformers architecture. Generally, these models are pre-trained with self-supervised learning on general tasks, and later, fine-tuned on specific tasks with small datasets of annotated documents.

The growing number of companies demanding Document AI services suggests that this is a growing sector. Companies such as Amazon, Google or Microsoft offer their data extraction services in the form of paid platforms hosted in the cloud. However, none of these solutions are focused on insurance policies, or require hundreds of manually annotated policies to function properly.

2. Project description

This thesis develops a tool for automatic extraction of data from insurance policies built with machine learning models that addresses the shortcomings of existing solutions. This tool must be able to extract: (1) all the text present in the document, (2) all related pairs

and (3) key elements defined by the user; all this without requiring user training. This tool is written in Python and executed on Google Colab notebook.

The AI models chosen for the development of the tool share several common characteristics. First, they are open-source models. Secondly, they must be downloadable ready to use, which implies that they must already be pre-trained and fine-tuned. Thirdly, they must offer good results in their respective tasks, if not SOTA. Fulfilling these characteristics, the LayoutLMv2 model is the central pillar of the tool and is used for labeling each token found in the text. This model, developed by Microsoft based on the BERT architecture, incorporates information about the text found in the document, its 2-D position and visual information about it. In addition, Google's PEGASUS model, fine-tuned on a dataset of SEC litigation releases and complaints, is also used. This model aims to summarize paragraphs of text found in the document, in order to present the information as organized and summarized as possible. In addition, the all-MiniLM-L6-v2 model is used to find the similarity between two sentences. This helps to extract key elements from the document. Finally, the OCR tool Pytesseract will be used to extract the text from the document.

3. Tool description

The data extraction process developed starts by importing the desired document. This document must contain a single page and be in image format. Figure 1 shows the architecture of the developed tool and, therefore, the analysis process that the introduced documents will follow.

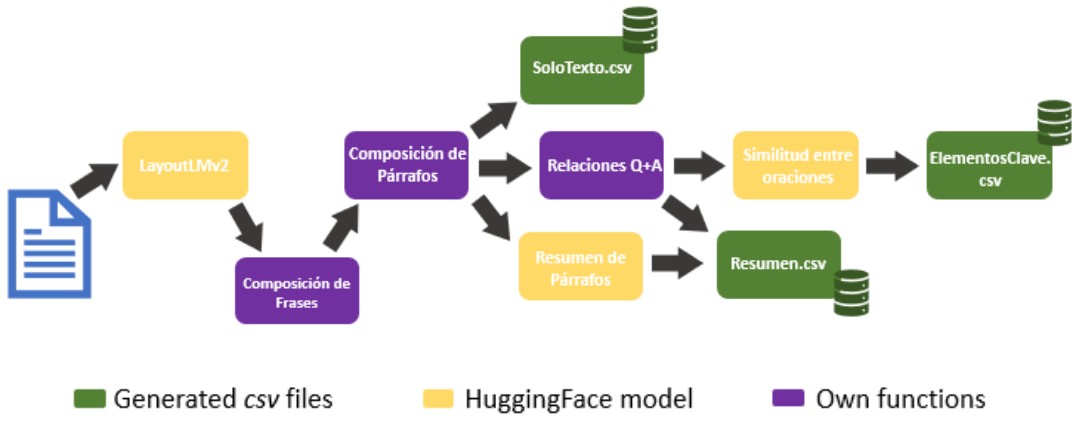

Figure 1. Architecture of the developed tool.

The document is first processed by LayoutLMv2, which will tag each token with one of the following tags {'Other', 'Header-Beginning', 'Header-Intermediate', 'Question-Beginning', 'Question-Intermediate', 'Answer-Beginning', Answer-Intermediate'}, as shown in Figure 2. These tags provide valuable information about the type of token and the possible relationships between them. However, LayoutLMv2 does not establish such relationships and labels words individually, but not sentences. Therefore, the next step is to form sentences from the tagged words.

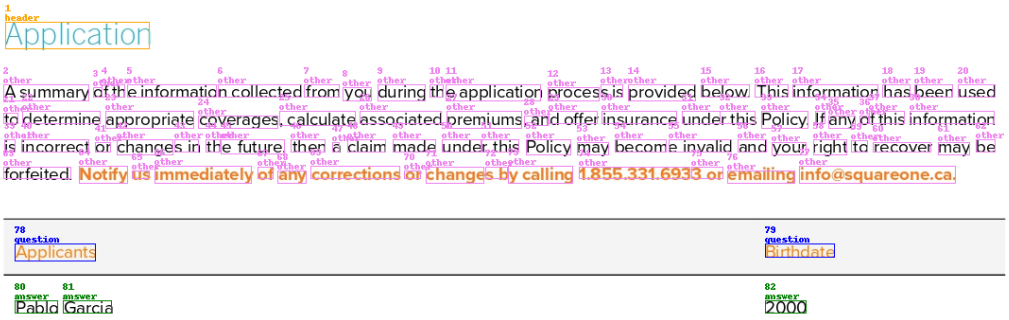

Figure 2. Example of token tagging a document with LayoutLMv2.

With this goal in mind, a function has been created that determines whether a word and the word to its right belong to the same sentence or not. Two words will belong to the same sentence when they are next to each other and they are of the same type {'Other', 'Header', 'Question', 'Answer'}. Once the sentences are formed, paragraphs are then formed following a very similar process. The paragraphs will be composed of sentences of the same type that are on top of each other. With the paragraphs already formed, we proceed first to extract the text from each of them with the OCR tool Pytesseract and save the text in the file *SoloTexto.csv*. This file will contain all the text found in the document, thus fulfilling the first proposed objective.

On the other hand, paragraphs of type 'Other' are processed by the legal-PEGASUS model, which summarizes them without any loss of relevant information. In parallel, the paragraphs, this time all of them, and not only those of type 'Other', are processed by a function that finds relations forming pairs with the format 'Question + Answers'. This function searches for certain structures based on the assumption that the answer associated with a question will be found to the right or below it. The relationships found by this function together with the summarized paragraphs of type 'Other' are saved in a new file called *Resumen.csv*, which, as its name suggests, serves as a summary of all the information contained in the document.

Finally, the tool analyzes the similarity between the questions of the 'Question + Answers' pairs and a series of dictionaries with elements that are intended to be extracted from the document. The tool incorporates by default three dictionaries: a generic dictionary for all insurance policies, a specific dictionary for car insurance and a specific dictionary for home insurance. However, the creation of new customized dictionaries is possible. For each item found in each of the dictionaries, the question with which it is most similar is selected. In this way, the tool is able to find the elements of the document that are most similar to each element that wants to be extracted from it. All these extracted elements are saved one last file called *ElementosClave.csv*.

4. Results

In this analysis of results, we will first analyze the results obtained by processing two different insurance policies with the tool developed in this thesis. Subsequently, these results will be compared with those obtained when analyzing the same policies with Google's Document AI tool.

In the first objective task, the extraction of all the text present in the document, the tool developed is able to correctly extract 89% of the sentences in one policy and 60% in the

other. Each policy has 82 and 77 sentences to be extracted, respectively. When forming relationships between document elements, the tool extracted 27 relationships from the first policy and 16 from the second. Table 1 shows part of the relationships extracted from the first policy.

QUESTION	ANSWER 1	ANSWER 2
Pollcy number	12345~67-89	
Effective	(01/04/2019 12:07 AM.	
Expiration	06/01/2019 12:01AM	
Named insured(s)	Jack Smith	Jane Smith
Underwritten by	Farmers Insurance	6301 Owensmouth Ave.
Policy premium + fees:	$798.48	
Veh. # year/make/model/vin	2019 Porsche Macan 4D 4wD-	
Bodily injury	$250K each person/$500K each incident	
Property damage	$100K each incident	Included
Medical coverage	$5,000 each person	$66.50
Deductible	3500	ACV
Collision	$1,000	ACY

Table 1. Relationships extracted by the tool.

Finally, in the key element extraction task, the tool extracted 12 elements out of the 14 target elements in the first policy and 2 out of 9 in the second policy. Table 2 shows some of the elements extracted from the first insurance policy.

GENERIC DICTIONARY – CAR INSURANCE POLICY	
Policy number	12345-67-89
Name insured	Jane Smith
Effective date	01/01/2019 0:01
Premium	3797.6
Address	1234 Main St
Limit	$250K each person/$800K each incident

Table 2. Key elements extracted by the tool.

On the other hand, Google's Document AI, using the generic document template, is able to perfectly extract the text by OCR, without making any errors. However, when forming relationships between elements, this tool only formed 18 relationships in the first policy and 2 in the second. Finally, the generic template used for the analysis of these policies does not offer the possibility of extracting key elements, and the creation of a customized template that does extract key elements would require at least 200 policies with the key elements manually annotated.

It can be concluded that the Google's tool incorporates a great OCR tool, but it has problems in forming relationships and is not able to extract key elements. In contrast, the tool developed in this thesis successfully establishes relationships between document elements and is able to extract a fair amount of the targeted key elements; however, it has problems in extracting the text accurately. It should be noted that this lack of precision when extracting text also worsens the results obtained in the rest of the tasks.

5. Conclusions

As a general conclusion, it has been possible to prove the effectiveness of models based on the Transformer architecture, trained with the pre-training plus fine-tuning method. These models stand out for their ability to understand human language and for the versatility of their applications. It has also been shown how introducing visual and positional elements increases the performance of the models in Document AI tasks. In addition, it has been observed in a practical way that an accurate OCR tool is a key element for the development of data extraction tools and should be given priority.

As future work to improve the tool presented in this thesis, it is advised, firstly, to replace LayoutLMv2 by the recently released LayoutLMv3, as the latter offers better results. Secondly, it may be interesting to adapt the tool to other languages, especially Spanish. This can be achieved by incorporating the LayoutXLM model, the multilingual version of the same family. Thirdly, improving the implementation and use of Pytesseract in this tool would be very beneficial and would significantly improve the results obtained. Finally, it would be desirable to develop a function that allows the analysis of multipage documents, as well as the analysis of documents by batches.

6. Referencias

Bao, H., Dong, L., & Wei, F. (2021). *BEiT: BERT Pre-Training of Image Transformers.* arXiv:2106.08254.

Cui, L., Xu, Y., Lv, T., & Wei, F. (2021). *Document AI: Benchmarks, Models and Applications.* arXiv:2111.08609v1.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transfromers for Languaje Understanding.* arXiv:1810.04805v2.

Huang, Y., Lv, T., Cui, L., Lu, Y., & Wei, F. (2022). *LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking.* arXiv:2204.08387.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). *Attention Is All You Need.* arXiv:1706.03762.

Walker, A. (2021). Document Searches Can Waste 100s of Hours. Obtenido de https://insurance-edge.net/2021/11/10/document-searches-can-waste-100s-of-hours/

Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., & Zhou, M. (2020). *MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers.* arXiv:2002.10957v2.

Xu, Y., Xu, Y., Lv, T., Cui, L., Wei, F., Wang, G., . . . Zhou, L. (2021). *LayoutLMv2: Multi-modal Pre-training for Rich Document Understanding.* arXiv:2012.14740v3.

Zhang, J., Zhao, Y., Saleh, M., & Liu, P. J. (2020). *PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization.* arXiv:1912.08777v3.

Índice de la memoria

Índice de figuras

Índice de tablas

Capítulo 1. INTRODUCCIÓN

1.1 MOTIVACIÓN DEL PROYECTO

Empresas de todo el mundo generan, envían y reciben diariamente enormes cantidades de documentos de todo tipo y con todo tipo de formatos. Lidiar de manera manual con tal cantidad de documentos puede tener un impacto negativo para las propias empresas. Por un lado, son millones los empleados que se ven enterrados en pilas documentos que deben leer de arriba abajo con el objetivo de extraer algún dato de ellos. De hecho, según un estudio de ABBYY, compañía líder en transformación digital a nivel internacional, un 27% de los trabajadores de Reino Unido manifestaron que perdían un día laboral entero por semana buscando información en documentos para satisfacer las necesidades de los clientes (Walker, 2021). Por otro lado, en muchos casos, los documentos generados por las empresas, en forma de recibos, emails, contratos, informes o planes de negocio, contienen información valiosa, que, de no ser procesada, quedará completamente archivada. Es por ello, que la extracción automatizada de datos de estos documentos juega un rol muy importante en la mejora del rendimiento de las empresas.

La extracción manual de información resulta demasiado ineficiente por la cantidad de tiempo que se emplea y su nula escalabilidad. Sin embargo, el *Document AI,* al combinar modelos de Inteligencia Artificial con la experiencia humana, ofrece una solución mucho más flexible y escalable. La automatización de este proceso permite el procesado de documentos de forma efectiva y veloz, quitándole a los trabajadores una importante carga de trabajo. De esta manera, no solo se ahorrará tiempo y dinero, también se mejorará la precisión de la extracción (Data Semantics, s.f.).

La creciente importancia de este sector puede observarse si analizamos el número de clientes que contratan los servicios que ofrecen las principales empresas en este sector. Según la plataforma web HG Insights (https://discovery.hgdata.com/), la herramienta Amazon Textract ha visto un incremento en el número de clientes de un 33% en el último año. Por su

parte, el número de clientes que utilizan la plataforma de Google Document AI ha incrementado en 100%. Por último, Microsoft Azure Form Recognition ha visto un incremento de un 46%. Estas cifras permiten ver que el *Document AI* es un sector al alza y que son más empresas cada año las que demandan estos servicios.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

En este segundo capítulo se define, en primer lugar, el concepto de *Document AI.* Así mismo, se describen las disciplinas del Procesamiento del Lenguaje Natural y la Visión Artificial; disciplinas de la Inteligencia Artificial que combinadas forman el *Document AI.* Más tarde, se resume la historia del *Document AI*, pasando después a describir las principales tareas de esta disciplina, los *datasets* considerados como *benchmarks* en el sector y los modelos que mejores resultados ofrecen en dichos *benchmarks.* Después se analizan las soluciones ya existentes, generalmente encontradas en forma de servicio de pago en la nube.

2.1 DEFINICIÓN DEL DOCUMENT AI

Document AI se refiere al proceso automatizado de comprensión, clasificación y extracción de información de documentos utilizando técnicas de Inteligencia Artificial (Cui, Xu, Lv, & Wei, 2021). El origen de dichos documentos es diverso, principalmente tratándose de documentos digitales, escaneados o capturas de pantalla de páginas web. El mayor reto para el *Document AI* es la creación de un modelo que presente buenos resultados para la gran variedad de diseños y formatos. Esta disciplina del ámbito de la Inteligencia Artificial combina otra dos: el Procesamiento del Leguaje Natural, o *Natural Language Processing (NLP)* en inglés, y la Visión por Ordenador, o *Computer Vision (CV).* Esta combinación de disciplinas resulta indispensable ya que los documentos generalmente son ricos tanto en texto como en elementos gráficos como títulos, figuras, listas o tablas, que le dan estilo y formato al documento. Por ello, es necesario combinar técnicas de ambas disciplinas, para que los modelos sean capaces de aprender a analizar texto y formato de manera conjunta.

En las siguientes dos secciones se definen las disciplinas de *NLP* y *CV*, y se describe de manera resumida qué modelos son comúnmente usados en cada una de ellas, en qué tareas se centra cada disciplina y qué aplicaciones tienen en el mundo real.

2.1.1 PROCESAMIENTO DEL LENGUAJE NATURAL

El Procesamiento de Lenguaje Natural se refiere a la rama de la Inteligencia Artificial que se preocupa de dotar a los ordenadores con la habilidad para entender el lenguaje, escrito y hablado (IBM Cloud Education, 2020b). El lenguaje humano está plagado de ambigüedades haciendo imposible la creación de reglas fijas manuales para su procesamiento y compresión. Por ello, la principal herramienta del *NLP* son los modelos de aprendizaje automático, ya que son más flexibles y capaces de generalizar. Los modelos de aprendizaje automático más utilizados hoy en día en esta rama de la Inteligencia Artificial están basados en Redes Neuronales Convolucionales, o *Convolutional Neural Networks (CNNs)* en inglés, Redes Neuronales Recurrentes, o *Recurrent Neural Networks (RNNs*), y los más recientes *Transformers*.

Para ayudar a los ordenadores a entender el lenguaje humano, el análisis del lenguaje ha sido dividido en distintas tareas parciales (IBM Cloud Education, 2020b):

- Reconocimiento del habla, o *Speech Recognition*, es la tarea que se encarga de convertir voz a texto. Esta tarea es en muchos casos el primer paso para poder analizar el lenguaje hablado.
- Etiquetado del discurso, o *Part of Speech Tagging*, es el proceso de identificar cada parte de un texto y etiquetarla según unas etiquetas predefinidas. Generalmente dichas etiquetas categorizan el texto como verbo, artículo, sustantivo, etc.
- Detección del sentido de las palabras, o *Word Sense Disambiguation*, es la tarea que se encarga de detectar el sentido real de palabras que tienen distintos significados según el contexto. Un simple ejemplo esto es la palabra ‘Banco’, cuyo significado dependerá del contexto en el que se encuentre.
- Reconocimiento de entidades con nombre, o *Named Entity Recognition*, es el proceso por el cual se identifican en el texto palabras o frases que correspondan a elementos comúnmente encontrados en el texto. Por ejemplo, ‘Pablo’ puede ser identificado como un nombre o ‘Madrid’ como un lugar.

- Correferencia, o *Co-reference Resolution* en inglés, es la tarea que se encarga de identificar cuando dos palabras presentes en el texto se refieren al mismo sujeto. Un ejemplo común es cuando las palabras 'Pablo' y 'él' se refieren al mismo sujeto.
- Análisis del sentimiento, o *Sentiment Analysis*, intenta analizar características subjetivas del texto, como las emociones. Un simple ejemplo de esta tarea es un modelo que otorga a cada fragmento de texto una puntuación del 0 al 1 según lo positivo que considere que es el texto.
- Generación de texto natural, o *Natural Language Generation*, se encarga de generar texto con apariencia de lenguaje humano. Esta es una de las tareas más importantes de *NLP* ya que es la usa para crear modelos que traducen de un idioma a otros, herramientas que resumen fragmentos de texto o modelos que responde a preguntas.

Son numerosas las aplicaciones que el Procesamiento del Lenguaje Natural tiene en el día a día. A continuación, se mencionan varias aplicaciones (IBM Cloud Education, 2020b).

- Detección de spam. La principal herramienta que tiene los proveedores de servicios de email para detectar si un correo electrónico es spam es un modelo de *NLP* que etiqueta cada correo electrónico como spam o no.
- Traductores de idiomas. Los traductores de idiomas como Google Translate o DeepL utilizan enormes modelos del lenguaje como motor principal.
- Los agentes virtuales y *chatbots* que podemos encontrar en numerosas páginas webs o productos de domótica también utilizan modelos de Procesamiento del Lenguaje Natural.
- El análisis de sentimiento en redes sociales tiene numerosas aplicaciones; desde detectar publicaciones que inciten al odio hasta inferir cual es el sentimiento del público hacia cierta acción bursátil.
- Las aplicaciones de resumir texto también son diversas y esta práctica se fundamenta en modelos del lenguaje. Sin ir más lejos, en este Trabajo de Fin de Grado se utilizará un modelo para resumir párrafos de texto encontrados en los documentos.

2.1.2 VISIÓN ARTIFICIAL

La visión artificial, o *Computer Vision* en inglés, es la rama de la Inteligencia Artificial que permite a ordenadores extraer información de elemento visuales como fotos o videos (IBM, 2021). Según IBM (2021) 'si la Inteligencia Artificial permite que las computadoras piensen, la visión por ordenador les permite ver, observar y comprender'. Dada la increíble complejidad del entorno y la gran diversidad de objetos que se encuentran a nuestro alrededor, para que un ordenador sea capaz de aprender y diferenciar imágenes y objetos es necesario proveer a los modelos con una gran cantidad de imágenes y videos. Los modelos de aprendizaje que se utilizan generalmente en esta rama de la Inteligencia Artificial están basados en Redes Neuronales Convolucionales.

De manera similar a como se hizo con el Procesamiento del Lenguaje Natural, para ayudar a los ordenadores a entender la complejidad del entorno se dividió la visión por ordenador en pequeñas subtareas parciales (Hmrishav, 2022).

- Clasificación de imágenes, o *Image Classification* en inglés, es la tarea encargada de clasificar imágenes según su contenido. Un ejemplo de esta tarea es clasificar documentos según su tipo: contratos, recibos, informes, emails, etc.
- La detección de objetos, o *Object Detection* en inglés, es el proceso que analiza imágenes y detecta distintos objetos dentro de ellas, como coches, personas, etc. Los modelos no solo deben ser capaces de detectar objetos, sino que también seguirlos a través de una secuencia de imágenes.
- Segmentación de imágenes, o *Image Segmentation*, se centra en dividir una imagen separando objetos unos de otros.
- La recuperación de imágenes basadas en contenido, *Content-based Image Retrieval*, se utiliza para encontrar imágenes dentro de grandes bases de datos basándose en su contenido. De esta manera se pueden encontrar imágenes similares a una dada.

Una vez más, las aplicaciones de esta rama de la Inteligencia Artificial son variadas y muy importantes en diversos sectores. A continuación, se proporcionan unos ejemplos para ilustrar la importancia de esta tecnología. Sin, embargo, sus aplicaciones son infinitas y cada

día surgen nuevas y novedosas maneras de aplicar dicha tecnología con el objetivo de optimizar procesos y ayudar al desarrollo humano (Hmrishav, 2022).

- En medicina es común el uso de la visión por ordenador para la detección de enfermedades como neumonía o cáncer a partir de imágenes tomadas en con en escáneres.
- En la industria manufacturera un ejemplo del uso de esta tecnología son los controles de calidad, pudiendo detectarse, mediante el análisis de imágenes, imperfecciones en piezas y productos.
- La visión por ordenador también es clave para que los vehículos autónomos funcionen correctamente, ya que durante la conducción es necesario detectar una gran cantidad de objetos para poder evitarlos y hacer la conducción segura.
- El reconocimiento óptico de caracteres, o *Optical Character Recognition (OCR)* en inglés, es una aplicación del *NLP* que tiene el objetivo de detectar y extraer el texto de una imagen (Amazon Web Services, s.f.).

2.2 HISTORIA DEL DOCUMENT AI

En las últimas décadas, el desarrollo del *Document AI* ha pasado por tres principales etapas; empezando por la utilización de métodos de extracción de datos basados en reglas fijas, pasando por la implementación de algoritmos de *Machine Learning*, y ahora, en el presente, el fuerte desarrollo de grandes modelos del lenguaje (Cui, Xu, Lv, & Wei, 2021).

Al principio de los años 90, se utilizaban principalmente métodos basados en reglas fijas para el análisis y entendimiento de documentos. Se observaban manualmente patrones en los documentos y se programaban reglas que extrajesen la información deseada. El principal problema de este método es que requería una gran cantidad de trabajo humano, para programar reglas para cada tipo de documento. Este método tampoco era escalable, ya que una pequeña modificación en el diseño del documento hacía que se tuviesen que programar nuevas reglas para poder extraer la información con precisión. Por ello, desde el año 2000, gracias al desarrollo del machine learning, se empezaron a adoptar métodos estadísticos que

detectaban patrones al analizar grandes cantidades de documentos anotados manualmente. Si es verdad que el empleo de machine learning mejoró los resultados obtenidos en comparación con los métodos anteriormente utilizados, el rendimiento seguía sin ser satisfactorio. El principal problema seguía siendo la escasa adaptabilidad que tenían estos modelos al cambiar el tipo de documento que se quería analizar, o su diseño. También, se requerían grandes cantidades de documentos que tenían que ser anotados manualmente, lo que una vez más, resultaba en un gran gasto de recursos, especialmente trabajo humano.

En los últimos años, con el fuerte desarrollo llevado a cabo en el campo del aprendizaje profundo[1], o *Deep Learning* en inglés, el *Document AI* ha entrado en una nueva era. La aparición de modelos como las Redes Neuronales Convolucionales, las Redes Neuronales de Grafos o los *Transformers* ha cambiado completamente el paradigma del análisis y extracción de datos de documentos. Gracias a estos modelos ya no son necesarias grandes cantidades de documentos anotados manualmente. Con este cambio de paradigma, los modelos son primero pre-entrenados en labores generales con un gran número de documentos sin anotar utilizando aprendizaje autosupervisado[2]. De esta manera, los modelos son capaces de adquirir una comprensión general del documento. Después, son afinados, o *fine-tuned* en inglés, con un pequeño *dataset* de documentos anotados para una tarea en concreto. Una vez afinado el modelo, estará listo para ser utilizado en nuevas predicciones. Este proceso de entrenamiento en dos etapas puede verse de manera gráfica en la Figura 2.1.

[1] El aprendizaje profundo es la rama del *Machine Learning* que se centra en la utilización de modelos compuestos por redes neuronales de 3 o más capas (IBM Cloud Education, 2020a).
[2] El aprendizaje autosupervisado es un método de aprendizaje automático en el que los propios datos cuentan de por sí con sus etiquetas de entrenamiento sin que hayan sido procesados etiquetados manualmente (LeCun, 2021).

Figura 2.1. Proceso de entrenado que involucra pre-entrenado y afinado.

A continuación, de describen las principales tareas que conforman la disciplina del *Document AI* (Cui, Xu, Lv, & Wei, 2021).

2.2.1 TAREAS DEL DOCUMENT AI

Análisis del diseño de un documento, o *Document Layout Analysis*, es el proceso de detección de patrones y elementos dentro de un documento, como párrafos de texto, tablas o figuras. Como se puede observar en la Figura 2.2, una vez detectados estos elementos los modelos devolverán un recuadro bordeando cada uno de ellos y una etiqueta informando de que tipo de elemento es.

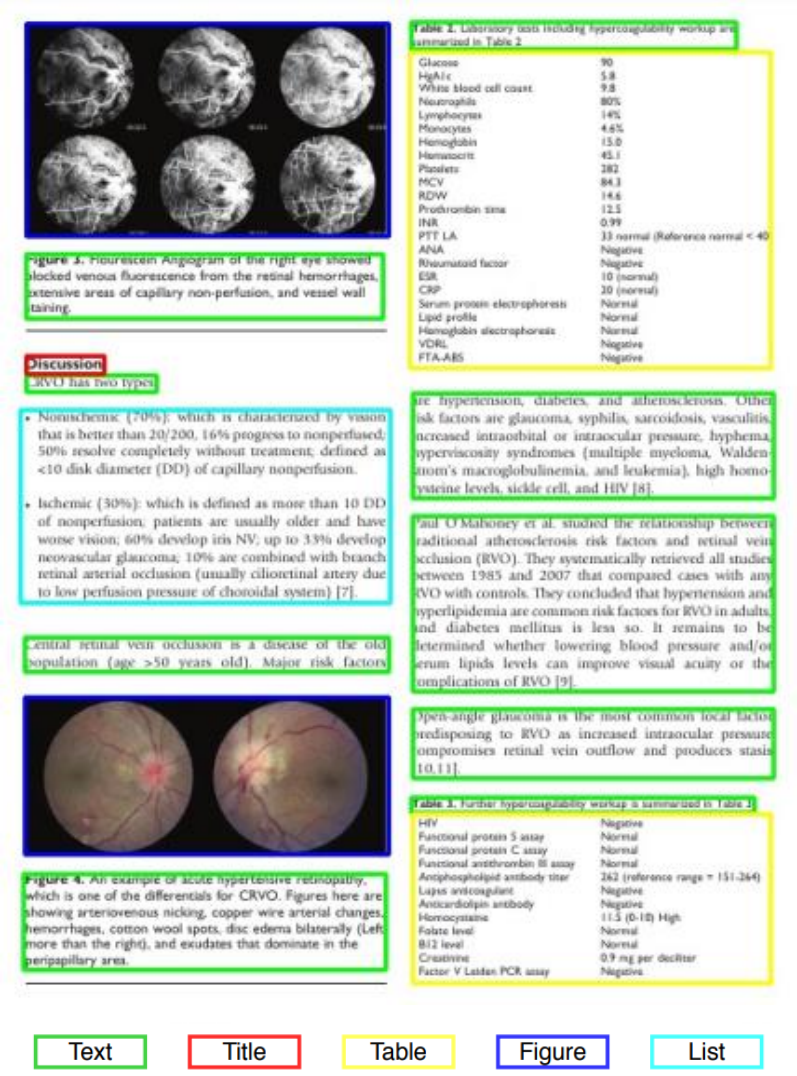

Figura 2.2. Ejemplo de documento con diseño anotado del dataset PubLayNet.

Imagen extraída de Zhong, Tang, & Jimeno (2019).

Extracción de información visual, o *Visual Information Extraction*, es la tarea encargada de la extracción de elementos o campos de texto determinados, como pueden ser nombres, lugares o fechas, entre otros. Generalmente los modelos han de ser entrenados para extraer información de un tipo de documento en concreto, con unos campos específicos y comunes a ese tipo de documento.

Respuesta a preguntas sobre documentos, o *Document Visual Question Answering*, tiene el objetivo de analizar el documento y ser capaz de responder a preguntas acerca del mismo. En la Figura 2.6 se muestra un ejemplo de esta tarea.

Clasificación de documentos, o *Document Image Classification*, se refiere al proceso de clasificar cada documento en categorías como recibos, documentos científicos, informes, etc.

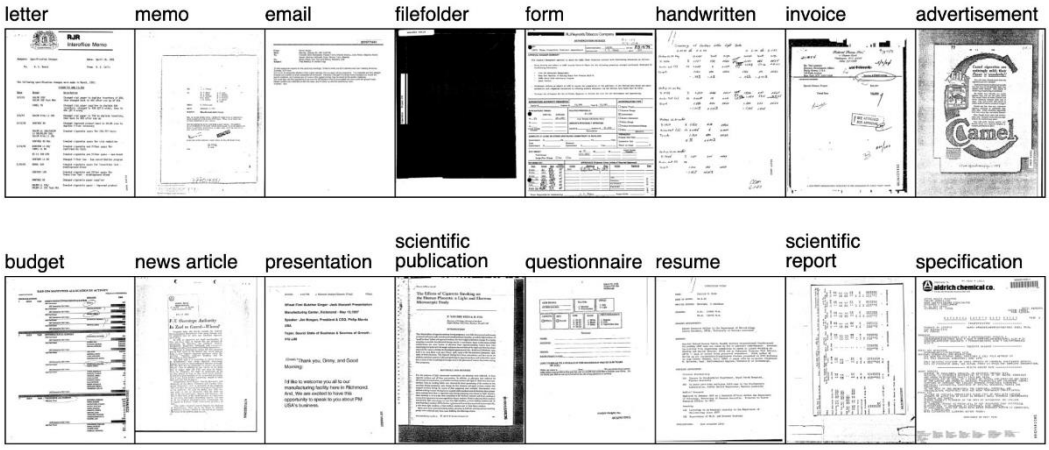

Figura 2.3. Ejemplo de documentos etiquetados según su categoría del dataset RVL-CDIP. Imagen extraída de https://adamharley.com/rvl-cdip/

Para cada una de estas tareas la comunidad científica ha establecido *dataset* que sirven como *benchmarks*, o referencia, con el objetivo de hacer la comparación del rendimiento de distintos modelos un proceso estandarizado.

2.2.2 BENCHMARKS

Existen numerosos *datasets* que pueden ser utilizados para cada una de las tareas. Muchos de ellos se muestran en la Tabla 2.1. De todos estos *benchmarks* los más utilizados por la comunidad científica en cada tareas son:

Document Layout Analysis

- PubLayNet (Zhong, Tang, & Jimeno, 2019). Este *dataset* contiene alrededor de 1 millón de artículos en formato PDF disponibles públicamente en PubMed Central. Cada uno de estos documentos ha sido anotado manualmente, etiquetando los distintos elementos. Un ejemplo del tipo de documentos y anotaciones que se pueden encontrar en este data queda reflejado en la Figura 2.2, extraído de este *dataset*. PubLayNet se considera el *dataset* de referencia a la hora de evaluar el rendimiento de un modelo en la detección y etiquetado del diseño de documentos.
- TableBank (Li, y otros, 2020). De manera similar a PubLayNet, TableBank es una *dataset* de documentos anotados, pero en este caso solo están anotadas las tablas. Consta de 417.234 tablas anotadas, cada una con su recuadro correspondiente. Este *dataset* se utiliza para evaluar la habilidad de un modelo para detectar tablas en documentos.

Visual Information Extraction

- *FUNSD*, acrónimo en ingles de *Form Understanding in Noisy Scanned Documents*, es un *dataset* de 199 formularios escaneados totalmente anotados, cada palabra o elemento del documento con una de las etiquetas (*question*, *answer*, *header* o *other*) (Jaume, Ekenel, & Thiran, 2019). Como puede verse en la Figura 2.4, la calidad de las imágenes de los documentos es pobre y contiene ruido, esto hace la tarea significativamente más difícil. Este *dataset* resulta de especial interés cuando se quiere entrenar un modelo que etiquete cada palabra encontrada en el documento. Estos modelos no extraerán información concreta de los documentos, pero si aportarán una visión general de la estructura del documento y las posibles relaciones entre cada elemento del mismo.

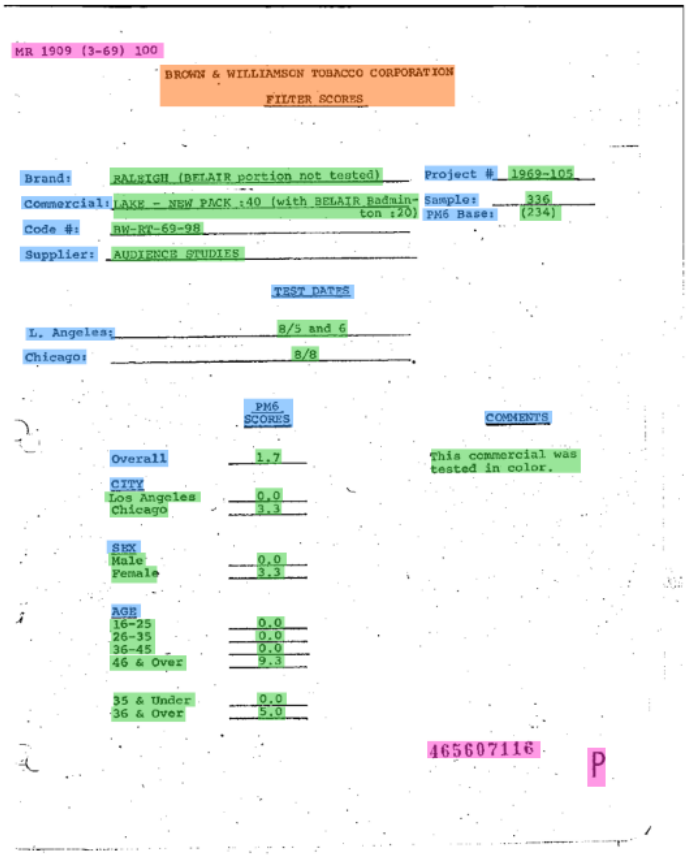

Figura 2.4. Ejemplo de documento totalmente anotado del dataset FUNSD. Imagen extraída de https://guillaumejaume.github.io/FUNSD/

- *CORD*, a *Consolidated Receipt Dataset for Post-OCR Parsing*, es un *dataset* que contiene más de 11.000 recibos de tiendas y restaurantes indonesios (Park, y otros, 2019). Cada uno de estas recibos está anotado según un total de 42 etiquetas como pueden ser, el nombre del menú, el importe total a pagar, la cantidad que corresponde a impuestos, cantidad pagada en efectivo, etc. A diferencia de FUNSD, este *dataset* se utiliza para entrenar y evaluar modelos en la tarea de extraer datos o campos determinados.

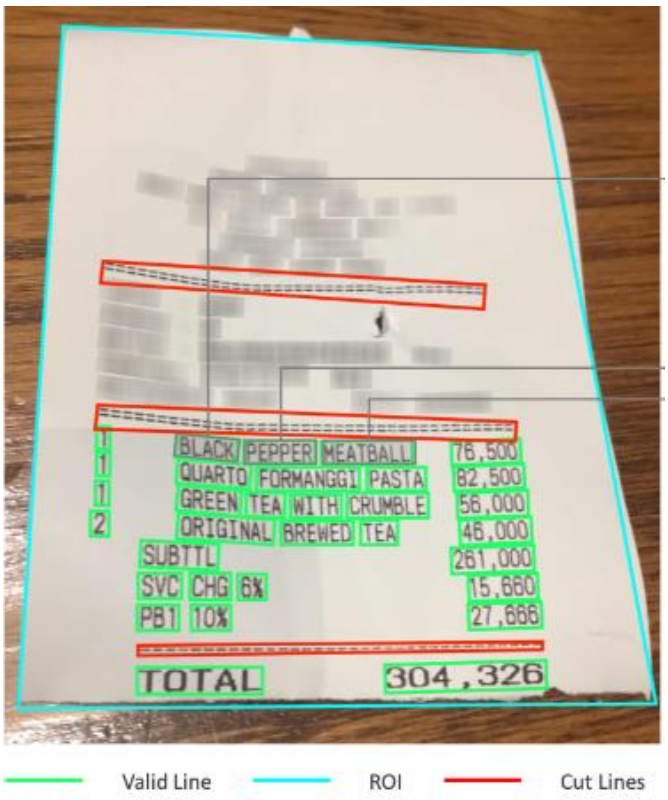

Figura 2.5. Ejemplo de recibo con campos anotados del dataset CORD. Imagen extraída de https://github.com/clovaai/cord

- *SROIE, ICDAR2019 Competition on Scanned Receipt OCR and Information Extraction*, es un *dataset* creado para el ICDAR[3] 2019 con el objetivo de fomentar el desarrollo de modelos para la extracción de datos a través de una competición (Huang, y otros, 2021). Este *dataset* contiene un total de 1.000 imágenes de recibos escaneados, cada uno de ellos conteniendo alrededor de 4 campos anotados. De igual manera que con CORD, el objetivo de este *dataset* es entrenar y evaluar modelos en la extracción de determinada información.

[3] ICDAR, *International Conference on Document Analysis and Recognition*, es el evento internacional para investigadores y profesionales de la comunidad del análisis de documentos más importante actualmente (ICDAR, s.f.).

Document Visual Question Answering

DocVQA es un *dataset* que contiene un total de 12.767 documentos variados de los que se han definido 50.000 preguntas con sus respuestas (Mathew, Karatzas, & Jawahar, 2021). Un claro ejemplo de uno de estos documentos puede verse en la Figura 2.6.

Figura 2.6. Ejemplo de documento del dataset DocVQA. Imagen extraída de

https://arxiv.org/pdf/2007.00398v3.pdf

Document Image Classification

RVL-CDIP es un *dataset* que consiste de 400.000 imágenes de documentos clasificados según 16 clases, entre las que se encuentran cartas, formularios, correos electrónicos, CVs, etc (Harley, Ufkes, & Derpanis, 2015). Como se puede ver en la Figura 2.3, estas imágenes tienen generalmente mala calidad y presencia de ruido.

Task	Benchmark	Langauge	Paper/Link
Document Layout Analysis	ICDAR 2013	En	Göbel et al. (2013)
	ICDAR 2019	En	Gao et al. (2019)
	ICDAR 2021	En	Yepes et al. (2021)
	UNLV	En	Shahab et al. (2010)
	Marmot	Zh/En	Fang et al. (2012)
	PubTabNet	En	Zhong et al. (2019a)
	PubLayNet	En	Zhong et al. (2019b)
	TableBank	En	Li et al. (2020a)
	DocBank	En	Li et al. (2020b)
	TNCR	En	Abdallah et al. (2021)
	TabLeX	En	Desai et al. (2021)
	PubTables	En	Smock et al. (2021)
	IIIT-AR-13K	En	Mondal et al. (2020)
	ReadingBank	En	Wang et al. (2021b)
Visual Information Extraction	SWDE	En	Hao et al. (2011)
	FUNSD	En	Guillaume Jaume (2019)
	SROIE	En	Huang et al. (2019)
	CORD	En	Park et al. (2019)
	EATEN	Zh	Guo et al. (2019)
	EPHOIE	Zh	Wang et al. (2021a)
	Deepform	En	Stray & Svetlichnaya (2020)
	Kleister	En	Stanisławek et al. (2021)
	XFUND	Zh/Ja/Es/ Fr/It/De/Pt	Xu et al. (2021b)
Document VQA	DocVQA	En	Mathew et al. (2021b)
	InfographicsVQA	En	Mathew et al. (2021a)
	VisualMRC	En	Tanaka et al. (2021)
	WebSRC	En	Chen et al. (2021)
	Insurance VQA	Zh	https://bit.ly/3602Vow
Document Image Classification	Tobacco-3482	En	Kumar et al. (2014)
	RVL-CDIP	En	Harley et al. (2015)

Tabla 2.1. Datasets más comunes para cada tarea del Document AI. Tabla extraída de Cui, Xu, Lv & Wei (2021).

2.3 ESTADO DEL ARTE: MODELOS

En este apartado se van a presentar de manera resumida los principales modelos que se consideran el estado del arte en *Document AI* en el momento de escribir este Trabajo de Fin de Grado. Para ello se va a hacer referencia al artículo científico *Document AI: Benchmarks, Models and Applications*, el cual recoge una revisión de los principales modelos que mejores resultados ofrecen en cada una de las tareas de esta disciplina. También se han extraído datos en forma de tablas de la página web *Papers With Code* (https://paperswithcode.com/), una web donde se puede encontrar información sobre todos los modelos que se consideran el estado del arte en la mayoría de las disciplinas de la Inteligencia Artificial. Por otro lado,

también se van a resumir los principales modelos y herramientas utilizadas en tareas de Procesamiento del Lenguaje Natural y Visión Artificial que van a ser utilizadas en el desarrollo de la herramienta propuesta en este TFG. Estas tareas son: (1) extracción de texto por OCR, (2) resumen de texto y (3) análisis de similitud entre oraciones.

2.3.1 MODELOS DEL DOCUMENT AI

2.3.1.1 Análisis del diseño de un documento, o Document Layout Analysis

La siguiente tabla recoge las precisiones medias de los modelos que mejor resultados obtienen al ser evaluados en el *dataset* PubLayNet.

Puesto	Modelo	Precisión Media	Año de publicación
1	LayoutLMv3	0,951	2022
2	DiT	0,949	2022
3	ResNext-101-32x8d	0,935	2018
4	DeiT	0,932	2020
5	BEiT	0,931	2021

Tabla 2.2. Ranking de modelos con mejores resultados en PubLayNet. Tabla extraída de https://paperswithcode.com/sota/document-layout-analysis-on-publaynet-val

En la tabla 2.3 se han anotado los modelos que ofrecen mejores resultados en las competiciones de detección de tablas de ICDAR. Hay que recalcar que se han unido en una misma tabla resultados de los *datasets* de ICDAR de 2013 y 2019.

Puesto	Modelo	F1 Media	Año de publicación
1	cascadetabnet	1,0*	2020
2	CDeCNet	1,0*	2020
3	TableNet	0,9662*	2020
4	DiT	0,9655**	2022

** Resultados obtenidos en ICDAR 2013*

*** Resultados obtenidos en ICDAR 2019*

Tabla 2.3. Ranking de modelos con mejores resultados en detección de tablas. Tabla extraía de https://paperswithcode.com/task/table-detection

De los resultados de las tablas 2.2 y 2.3 se puede observar que para la tarea de detectar y etiquetar distintos elementos del diseño de un documento los modelos pre-entrenados de propósito general basados en *Transformers*, como LayoutLMv3 o DiT, ofrecen mejores resultados; mientras que, para una tarea menos global, como es la detección únicamente de tablas, los modelos de propósito único basados en Redes Neuronales Convolucionales, como cascadetabnet o CDeCNet, proporcionan mejores resultados.

2.3.1.2 Extracción de información visual, o Visual Information Extraction

En el *dataset* de FUNSD los modelos que mejor resultados ofrecen están listados en la siguiente tabla.

Puesto	Modelos	F1 (%)	Año de publicación
1	LayoutLMv3	92,08	2022
2	LiLT	88,41	2022
3	DocFormer	84,55	2021
4	LayoutLMv2	84,2	2020

Tabla 2.4. Ranking de modelos con mejores resultados en el dataset FUNSD. Tabla extraída de https://paperswithcode.com/sota/semantic-entity-labeling-on-funsd

En la siguiente tabla se muestran en orden los modelos que con los mejores resultados en el *dataset* CORD.

Puesto	Modelos	F1 (%)	Año de publicación
1	LayoutLMv3	97,46	2022
2	DocFormer	96,99	2021
3	TILT	96,33	2021
4	LiLT	96.07	2022
5	LayoutLMv2	96,01	2020

Tabla 2.5. Ranking de modelos con mejores resultados en el dataset CORD. Tabla extraída de https://paperswithcode.com/sota/key-information-extraction-on-cord

Por último, se muestra la Tabla 2.6 con el ranking de modelos con mejores resultados en el *dataset* SROIE.

Puesto	Modelos	F1 (%)	Año de publicación
1	TILT	98,10	2021
2	LayoutLMv2	97,81	2020

Tabla 2.6. Ranking de modelos con mejores resultados en el dataset SROIE. Tabla extraída de https://paperswithcode.com/sota/key-information-extraction-on-sroie

Los rankings de estos tres *datasets* confirman que para las tareas de extracción de información visual los modelos basados en *Transformers*. Esto confirma la efectividad de esta arquitectura a la hora de adquirir conocimiento acerca del documento, gracias al proceso de pre-entrenado más afinado.

2.3.1.3 Respuesta a preguntas sobre documentos, o Document Visual Question Answering

Para la tarea de *Visual Question Answering* y en concreto el *dataset* de DocVQA, los modelos que mejores resultados obtienen son los presentes en la siguiente tabla.

Puesto	Modelos	ANLS	Año de publicación
1	TILT	0,8705	2021
2	LayoutLMv2	0,8348	2020
3	LayoutLMv3	0,8337	2022

Tabla 2.7. Ranking de modelos con mejores resultados en el dataset DocVQA. Tabla extraída de https://paperswithcode.com/sota/visual-question-answering-on-docvqa-test.

Una vez más, encontramos en la cabeza del ranking modelos que ya encabezaban los rankings en otras tareas diferentes. Puede verse así la versatilidad y flexibilidad de los modelos basados en *Transformers*.

2.3.1.4 Clasificación de documentos, o Document Image Classification

Para esta última tarea, se muestran a continuación los modelos con mayor precisión en el *dataset* RVL-CDIP.

Puesto	Modelo	Precisión media (%)	Año de publicación
1	DocFormer	96,17	2021
2	LayoutLMv3	95,93	2022
3	LiLT	95,68	2022
4	LayoutLMv2	95,64	2020
5	TILT	95,52	2021

Tabla 2.8. Ranking de modelos con mejores resultados en el dataset RVL-CDIP.

Tabla extraída de https://paperswithcode.com/sota/document-image-classification-on-rvl-cdip.

En la tarea de clasificación de documentos los modelos basados en la arquitectura *Transformer* también encabezan el ranking.

2.3.2 MODELOS DEL PROCESAMIENTO DEL LENGUAJE NATURAL

Para el desarrollo de la herramienta de extracción de datos de este TFG, del campo del Procesamiento del Lenguaje Natural, se van a utilizar solamente modelos enfocados en dos tareas: resumir texto y encontrar la similitud entre dos oraciones. Es por ello, que la revisión de los modelos considerados el estado del arte en este campo se hará solo para estas dos tareas, ya que una revisión de todas las tareas recogidas en el ámbito del Procesamiento del Lenguaje Natural daría por sí sola para un Trabajo de Fin de Grado.

2.3.2.1 Text Summarization, o Resumen de texto

En la tarea de resumir texto es importante tener que el tipo de texto que se desea resumir y el *dataset* con el que el modelo fue entrenado sean lo más parecidos posibles. Esto es de especial importancia al resumir textos legales, ya que estos textos cuentan con un

vocabulario específico y están cargados de ambigüedades y referencias a otros documentos (Kanapala, Pal, & Pamula, 2019).

Existen diversos *datasets* utilizados como *benchmarks* para esta tarea. Cada uno de estos *datasets* está formado por texto de diversos orígenes, como puede ser emails, artículos periodísticos, texto encontrados en webs, entre otros. Estos *datasets* cuentan con cientos de miles, e incluso millones, de textos y sus correspondientes resúmenes. El *dataset* CNN/Daily Mail es uno de los más utilizados en la industria a la hora de evaluar modelos de resumen de texto. A continuación, se muestran los modelos que mejores resultados obtienen en dicho *dataset.*

Puesto	Modelo	Rouge-1 (%)	Año de publicación
1	BRIO	47,78	2022
2	BART + SimCLS	46,67	2021
3	GLM-XXLarge	44,7	2021
4	BART + R-Drop	44,51	2021
(…) 12	Pegasus	44,17	2020

Tabla 2.9. Ranking de modelos con mejores resultados en CNN/Daily Mail Dataset.

Tabla extraída de https://paperswithcode.com/sota/abstractive-text-summarization-on-cnn-daily.

Una vez más, los modelos que mejores resultados ofrecen están basados en Transformers. También se puede observar que los resultados ofrecidos por los modelos que encabezan el ranking son muy similares, especialmente a partir del tercer puesto.

En cuanto al caso de resumen de textos legales, no se ha encontrado ningún *dataset* que sea utilizado como *benchmark* en la industria.

2.3.2.2 Sentence Similarity, o Similitud entre oraciones

En la tarea de analizar la similitud entre dos oraciones, entre todos los *datasets*, se ha elegido para la comparativa el *dataset* STS, *Semantic Textual Similarity*.

Puesto	Modelo	Pearson Correlation	Año de publicación
1	SMART-RoBERTa	0,929	2019
2	StructBERTRoBERTa	0,928	2019
3	Mnet-Sim	0,927	2021
4	T5-11B	0,925	2019
5	ALBERT	0,925	2019

Tabla 2.10. Ranking de modelos con mejores resultados en CNN/Daily Mail Dataset.

Tabla extraída de https://paperswithcode.com/sota/semantic-textual-similarity-on-sts-benchmark.

En la tarea de similitud entre oraciones, los modelos basados en *Transformers* también encabezan el ranking.

Por otro lado, en la plataforma de HuggingFace (https://huggingface.co/), el modelo más descargado en esta tarea es *all-MiniLM-L6-v2*, con más de 3,4 millones de descargas en el último mes.

2.3.3 MODELOS DE VISIÓN ARTIFICIAL

2.3.3.1 OCR

Cómo ya se ha mencionado anteriormente, para extraer el texto contenido en documentos es necesario disponer de una herramienta OCR. Son numerosas las soluciones de código abierto que existen, sin embargo, resulta difícil establecer un ranking en función de su rendimiento. Es por ello, que se van a considerar aquellas herramientas de OCR con mayor popularidad. Entre ellas destacan: Pytesseract y OpenCV.

2.4 ESTADO DEL ARTE: SOLUCIONES EXISTENTES

En la actualidad, son principalmente las grandes empresas tecnológicas, como Amazon, Google o Microsoft, las que ofrecen soluciones de *Document AI*; generalmente en forma de servicios de pago. También existen pequeñas empresas que ofrecen servicios similares. Sin embargo, el análisis de las soluciones existentes se va a centrar en los productos ofrecidos por las grandes tecnológicas, ya que, al fin y al cabo, son ellas quienes impulsan el desarrollo de este campo.

2.4.1 AMAZON TEXTRACT

Amazon dispone de su servicio Textract (https://aws.amazon.com/es/textract/). Según la información de su web oficial, Textract es un servicio de *Machine Learning* que extrae texto y datos estructurados, como tablas y formularios, sin ningún tipo de configuración o plantillas. En concreto, los servicios que ofrece son:

- Extracción de texto mediante OCR.
- Extracción de pares clave-valor de formularios.
- Extracción de tablas.
- Respuesta a preguntas sobre el documento.
- Extracción de datos clave de facturas y recibos.
- Extracción de datos de documentos de identidad.

Por otro lado, estos servicios tienen características interesantes como que el usuario tenga la posibilidad de editar los umbrales de confianza de los modelos o posibilidad de agregar revisiones humanas gracias a Amazon Augmented AI, para supervisar los modelos.

2.4.2 GOOGLE DOCUMENT AI

Google ofrece su servicio llamado Document AI (https://cloud.google.com/document-ai/). La empresa lo define la plataforma como ‘una consola unificada para el procesamiento de documentos’. Document AI permite:

- Extracción de texto mediante OCR.
- Extracción de pares clave-valor de formularios.
- Extracción de tablas.
- Extracción de datos clave (solo para determinados documentos).

Los servicios de extracción de texto, extracción de pares clave-valor y extracción de tablas puede ser utilizados para todo tipo de documentos. Para mejorar los resultados la plataforma te permite seleccionar el tipo de documento deseado, usar un procesador general o incluso, crear tu propia plantilla, gracias a AutoML. La extracción de datos clave solo es válida para cierto tipo de documentos, entre los que se encuentran: contratos, documentos de identidad y varios formularios gubernamentales de EEUU.

2.4.3 MICROSOFT AZURE FORM RECOGNIZER

Microsoft por su parte, dispone de Azure Form Recognizer (https://docs.microsoft.com/en-us/azure/applied-ai-services/form-recognizer/), un servicio de *Machine Learning* para el análisis de formularios y documentos. Esta herramienta permite:

- Extracción de texto.
- Extracción de tablas.
- Análisis del diseño de un documento.
- Extracción de pares clave-valor.
- Extracción de datos clave

Azure Form Recognizer incorpora modelos predeterminados para la extracción de datos de documentos como: recibos, formularios gubernamentales, documentos de identidad, entre otros. También permite la creación de modelos personalizados, debiendo proveerle también con numerosos documentos anotados manualmente.

Capítulo 3. DEFINICIÓN DEL TRABAJO

En este tercer capítulo se presenta la solución propuesta en este Trabajo de Fin de Grado, una herramienta de extracción automática de datos de pólizas de seguros construida a partir de modelos de aprendizaje automático.

En primer lugar, se justifica el desarrollo de dicha herramienta, apoyándose en las carencias de las soluciones existentes. En segundo lugar, se presentan los objetivos que se han planteado para de dicha herramienta. A continuación, se describen las limitaciones de recursos existentes a la hora de desarrollar la herramienta. Más tarde, se analiza la estructura de una póliza de seguros, detallando los tipos de seguros más comúnmente contratados por los hogares españoles y los elementos más comunes encontrados en cada tipo de seguro. Finalmente, se describirán las herramientas y modelos de Inteligencia Artificial utilizados, justificando la elección de cada una de ellas.

3.1 JUSTIFICACIÓN

Una vez entendido el contexto del *Document AI* y analizadas las soluciones existentes para la extracción de datos de pólizas de seguros queda justificado el desarrollo de una nueva herramienta de *Document AI* centrada en pólizas de seguros por los siguientes argumentos:

- Las soluciones actuales no están centradas en pólizas de seguros. Siendo verdad que empresas como Amazon, Google o Microsoft ofrecen servicios de extracción de datos de documentos, ninguno de estos servicios se centra específicamente en pólizas de seguros, por lo que los resultados que estas herramientas ofrecen no son aceptables para el uso empresarial.
- Las grandes tecnológicas ofrecen también la posibilidad de crear plantillas personalizadas para mejorar los resultados obtenidos extrayendo datos de documentos de los que no ofrecen plantillas predeterminadas. El problema con la creación de estas plantillas es que requieren un gran número de documentos

anotados. En la documentación de Google Cloud, se aconseja que para cada campo que se quiera extraer se aporten al menos 200 documentos en los que aparezca ese campo anotado. El problema es que la anotación de estos documentos puede ser muy tediosa para el usuario, e incluso existe la posibilidad de que no disponga de los cientos de ejemplares de pólizas de seguros que se necesitan.

3.2 OBJETIVOS

El objetivo principal de este TFG es la creación de una herramienta de Inteligencia Artificial para la extracción automática de datos de pólizas de seguros.

Desde una perspectiva técnica, la herramienta no debe de necesitar ningún tipo de entrenamiento, debe exigir el mínimo ajuste por parte del usuario y debe ser capaz de extraer: (1) todo el texto presente en el documento, (2) todos los pares de datos relacionados y (3) elementos clave definidos por el usuario. Toda esta información debe ser presentada de la manera más visual y resumida posible. De requerir la herramienta ajuste por parte del usuario la manera en la que el usuario realizará los ajustes deberá ser lo más simple e intuitiva posible, de forma que cualquier usuario sin ningún tipo de experiencia de programación sea capaz de utilizar esta herramienta. La herramienta debe ser flexible y tener la capacidad de adaptarse a cualquier tipo de seguro, e incluso de documento. Estos objetivos se verán abordados en los apartados subsiguientes y en la sección de 'Herramienta desarrollada'.

Desde un punto de vista financiero, la herramienta deberá ser viable económicamente. Por ello, es importante que su implementación no sea costosa para el usuario. Esto se discutirá más adelante en la sección de 'Viabilidad Económica'.

Finalmente, desde una perspectiva social, la herramienta deberá estar alineada con los Objetivos de Desarrollo Sostenible (ODS). Este punto se abordará en la sección 'Alineación con los Objetivos de Desarrollo Sostenible'.

3.3 LIMITACIONES

En esta sección se van a discutir las limitaciones existentes a la hora de desarrollar la herramienta propuesta en este Trabajo de Fin de Grado.

- En primer lugar, como ya se ha comentado en la sección de 'Descripción de las Tecnologías', para afinar los modelos en el análisis de un tipo de documento en concreto es necesario disponer de un *dataset* de documentos anotados. Debido a las grandes dificultades a la hora de conseguir ejemplos de pólizas de seguros, no se dispone de ningún *dataset* de pólizas de seguro anotadas. Esto implica que se deberán descartar para el desarrollo de este TFG los modelos de Inteligencia Artificial que no se puedan descargar pre-entrenados.
- En segundo lugar, el entrenamiento de grandes modelos, que cuentan con cientos de millones de parámetros, resulta muy caro a nivel computacional, haciendo falta equipos especializados que no se disponen durante el desarrollo de este TFG. Es por ello, que las operaciones que exijan un gran coste computacional, como el entrenamiento de grandes modelos, han de ser descartas.
- En tercer lugar, el código de muchos modelos considerados el estado de arte no ha sido liberado para el uso público. Es por ello, que se recurrirá a plataformas como Hugging Face, donde se pueden encontrar decenas de miles de modelos disponibles para el uso público.

3.4 ESTRUCTURA DE UNA PÓLIZA DE SEGUROS

Las pólizas de seguros son un documento de gran complejidad, ya que combinan párrafos de texto, tablas, listas y figuras. También añade complejidad el hecho de que el lenguaje empleado sea lenguaje de estilo jurídico, rico en términos específicos del sector. A todo esto, se le suma que cada empresa utiliza un formato y diseño totalmente distinto. El formato y diseño no solo cambian de empresa a empresa, sino que también dentro de una misma empresa al tratarse de un tipo de seguro u otro. Es por ello, que herramientas de extracción de datos de documentos de tipo genérico no dan resultados satisfactorios. Como ya se ha

mencionado, en este TFG se va a desarrollar una herramienta de extracción automática de datos de pólizas de seguros. Para ello, es necesario tener un buen entendimiento de la estructura de una póliza de seguros y de la naturaleza de los datos que se desean extraer.

Figura 3.1. Diagrama de los tipos de seguros.

En primer lugar, es importante saber qué tipo de seguro es el documento en cuestión, para tener una idea del vocabulario específico que va a contener la póliza. Según un artículo publicado en la web de Rankia, los seguros se pueden agrupar en tres grandes grupos según su tipo: seguros personales, seguros de bienes y seguros de servicios (Grafiada, 2022). Dentro de los seguros personales encontramos los seguros de salud, de vida, de accidentes personales y de dependencia, como bien se puede ver en la Figura 3.1. Por otro lado, dentro de los seguros de bienes, encontramos los seguros de automóvil, del hogar, de robo y de responsabilidad civil, entre otros. Por último, dentro del grupo de los seguros de servicios encontramos los seguros de asistencia en viaje, de decesos y de defensa jurídica. Cabe mencionar, que todas estas categorías de seguros mencionadas no cubren todo el espectro de tipos de seguros, y que existen variedad de seguros dentro de cada una de estas tipologías, en función de las necesidades del cliente.

De todos los tipos de seguros mencionados, en España, según la Encuesta de Presupuestos Familiares (EPF) de 2018 realizada por el Instituto Nacional de Estadística (INE), los tipos de seguros más comunes en los hogares españoles son los mostrados en la Figura 3.2. Encabeza la lista el seguro de automóvil, siendo éste contratado por un 79% de los hogares; le sigue el seguro de hogar, contratado por un 72%; y en el tercer puesto se encuentra el seguro de decesos, contratado en un 45% de los hogares (Unespa, 2018).

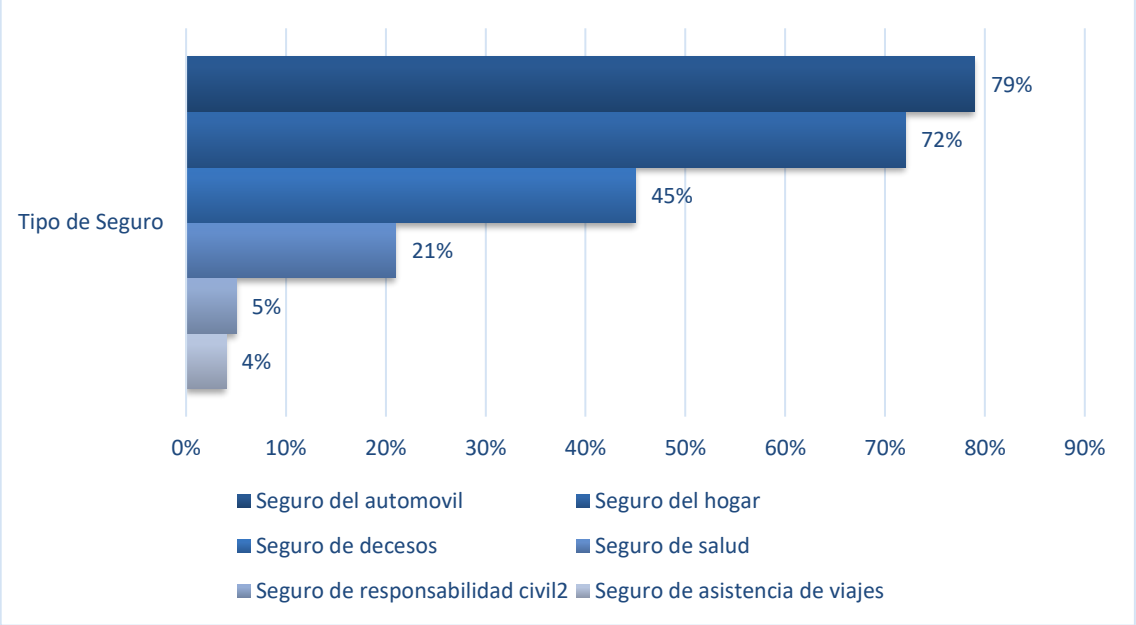

Figura 3.2. Tipos de seguros más comunes en los hogares españoles. Datos extraídos de (Unespa, 2018).

A parte de los tipos de pólizas de seguro es importante saber que elementos conforman dichas pólizas. Esta información resulta de especial interés a la hora de extraer elementos clave del documento como el nombre del asegurado o de la empresa aseguradora. A continuación, se muestra una lista con los elementos más comunes presentes en las pólizas de seguros independientemente de su tipo:

- Nombre del asegurado
- Aseguradora
- Número de póliza

- Fecha efectiva
- Dirección del asegurado
- Prima del seguro

Dependiendo del tipo de seguro encontraremos terminología específica. A continuación, se muestra una lista con los elementos más comunes encontrados en pólizas de seguros del automóvil y otra lista con los elementos más comunes en pólizas de seguros del hogar.

Seguro del automóvil:

- Modelo del vehículo
- Deducible por colisión
- Cobertura por daños corporales
- Cobertura por daños materiales
- Cobertura médica
- Cobertura por daños corporales por motorista no asegurado

Seguro del hogar:

- Cobertura por daños a la vivienda
- Cobertura por daños a otras estructuras
- Cobertura por daños a bienes personales
- Cobertura por pérdida de uso
- Cobertura por Responsabilidad civil

Aclarar que estos términos han sido sacados de pólizas de seguros en inglés y después traducidos a español, por lo que es posible que no coincidan exactamente con la terminología utilizada es España.

3.5 HERRAMIENTAS UTILIZADAS

En esta sección se van a discutir las herramienta utilizadas durante el desarrollo de este TFG, justificando cada una de ellas.

3.5.1 PYTHON

La herramienta desarrollada en este TFG está escrita íntegramente en Python. Python tiene un gran número de ventajas a la hora de desarrollar programas de Inteligencia Artificial. En primer lugar, y quizás el argumento más importante, es que Python cuenta con un gran número de librerías que hacen más rápido y sencillo el desarrollo de programas y

herramientas. Para este TFG resultan de especial interés las librerías especializadas en modelos de Inteligencia Artificial, como PyTorch, o Tensorflow, así como inmensidad de librerías de manejo y operaciones con vectores, como Numpy. En segundo lugar, Python es uno de los lenguajes de programación más populares que existen, siendo en 2020 por tercer año consecutivo el lenguaje de programación más popular según el ranking realizado por IEEE (IEEE, 2020). Esto implica que Python cuenta con una gran comunidad y numerosas entradas en foros como Stackoverflow resolviendo dudas y errores comunes, que resultan de gran utilidad a la hora de desarrollar una programa.

3.5.2 GOOGLE COLAB

Google Colab es un producto gratuito de Google que permite a cualquier usuario escribir y ejecutar código de Python en el navegador. Como menciona la propia empresa 'es especialmente adecuado para tareas de aprendizaje automático y análisis de datos' ya que ofrece acceso sin coste a GPUs. El acceso a una GPU permite que el entrenado y la ejecución de los modelos de aprendizaje automático sea mucho más rápida. Como se puede ver en la Figura 3.3, el tiempo empleado para el entrenamiento de un modelo en el *dataset* Fashion-MNIST utilizando la GPU de Colab, es sensiblemente menor que el tiempo empleado por ordenadores personales, algunos incluso dotados de sus propias GPUs, como es el caso del Lenovo Legion Y540 (Radečić, 2020).

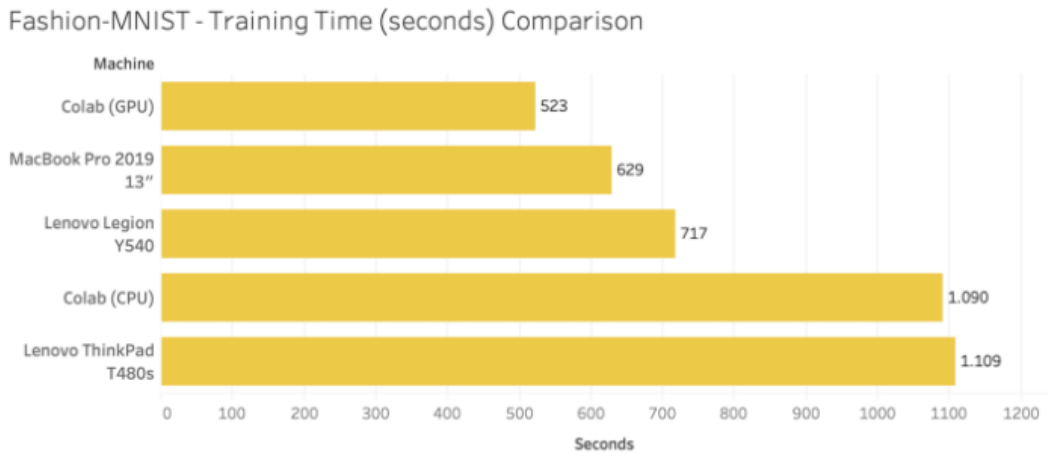

Figura 3.3. Comparativa del tiempo empleado por distintos ordenadores en entrenar un modelo. Gráfica extraída de (Radečić, 2020).

3.5.3 GITHUB

GitHub es una plataforma alojada en la nube que incorpora Git, un sistema de control de versiones. Esta plataforma permite a sus usuarios alojar su código en la nube y colaborar en el desarrollo de programas manteniendo un seguimiento detallado de las modificaciones realizadas. Resulta de especial interés la posibilidad de sincronizarla con Google Colab, pudiendo así alojar en un repositorio de GitHub los cuadernos de Colab.

3.5.4 HUGGING FACE

Hugging Face es una comunidad y plataforma de Inteligencia Artificial que permite a sus usuarios crear, entrenar y utilizar modelos de Inteligencia Artificial de código abierto. Su objetivo principal es conseguir una colaboración fluida y abierta entre los miembros de la comunidad para intentar 'resolver la Inteligencia Artificial'. En ella se pueden encontrar cerca de 60.000 modelos ya entrenados considerados el estado del arte, especialmente en ámbitos como el Procesamiento del Lenguaje Natural. También alberga cerca de 6.000 *datasets*, disponibles para uso público. Numerosas empresas líderes en el sector, como Facebook, Google, Amazon o Microsoft, comparten aquí sus modelos, depositando su confianza en la plataforma. Gran parte de los modelos mencionados en el análisis del estado de arte están publicados en Hugging Face.

3.6 MODELOS UTILIZADOS

Una vez hecho el análisis y la comparativa de los modelos considerados el estado del arte del *Document AI*, y teniendo en cuenta las limitaciones existentes y los objetivos propuestos, se ha decidido utilizar como espina dorsal de la herramienta desarrollada este Trabajo de Fin de Grado, el modelo LayoutLMv2. También son necesarios un modelo para resumir texto y otro para hallar la similitud entre dos oraciones. Para resumir texto se ha elegido legal-PEGASUS y para similitud entre oraciones all-MiniLM-L6-v2. Por último, la elección de una buena herramienta de OCR resulta imprescindible para el desarrollo de este TFG. En este caso, se ha escogido Pytesseract como OCR. En los párrafos siguientes se justifica la

elección de cada una de estas herramientas. Más adelante se detalla la arquitectura de cada uno de estos modelos y su utilidad en esta herramienta.

En primer lugar, la elección de LayoutLMv2 como pilar central de la solución propuesta en este TFG radica en que, de los modelos que encabezan los rankings de resultados de las tares de *Document AI* , LayoutLMv2 es el único modelo que cumple con todas las características necesarias. El resto de los modelos no cumplen alguno de los siguiente requisitos:

- Ofrecer buenos resultados en las principales tareas del *Document AI*.
- Aportar información valiosa acerca del documento.
- Ser de código abierto.
- Poder ser descargado listo para el uso, lo que implica que ya esté pre-entrenado y afinado.

Cumpliendo con todos estos requisitos, se ha escogido LayoutLMv2 que ofrece los siguientes resultados en las principales tareas de *Document AI*: FUNSD (0,8420), CORD (0,9601), SROIE (0,9781), DocVQA (0,8672), RVL-CDIP (0,9564). El modelo también aporta información muy valiosa sobre los documentos ya que etiqueta cada *token* de manera bastante detallada, aportando información no solo del propio *token*, sino también de posibles relaciones con otros *tokens*. Por último, se puede encontrar una versión de LayoutLMv2 pre-entrenada y afinada en el dataset FUNSD en Hugging Face.

Cabe remarcar que el modelo de la misma familia, LayoutLMv3, también cumple estas mismas características y presenta mejores resultados que su versión anterior en la mayoría de los *benchmarks*. Sin embargo, en el momento de escribir esta memoria la herramienta de extracción de datos propuesta en este TFG ya ha sido desarrollada utilizando el modelo LayoutLMv2. Es por ello, que se dejará la posible implementación de LayoutLMv3 como trabajo futuro.

En segundo lugar, para la tarea de resumir texto, se ha buscado un modelo que estuviese enfocado al resumen de textos legales; ya que estos texto cuentan con un vocabulario muy técnico y un modelo entrenado con fuentes de texto no legales podría ofrecer peores resultados. Es por ello, que el modelo escogido para esta tarea debe:

- Ofrecer buenos resultados en la tarea de resumir texto.
- Haber sido afinado en un *dataset* de textos legales.
- Ser de código abierto.
- Ser descargado listo para el uso, lo que implica que ya esté pre-entrenado y afinado.

Por este motivo, cumpliendo con todos los requisitos, se ha escogido el modelo PEGASUS, de Google, pre-entrenado en el *dataset* CNN/Daily Mail y afinado en una colección más de 2700 de comunicados y reclamaciones de litigios de la SEC. Esta versión afinada del PEGASUS ha sido encontrada en Hugging Face y a su autoría pertenece al usuario *nsi319*.

En tercer lugar, como modelo para analizar la similitud entre dos frases se ha escogido el modelo all-MiniLM-L6-v2. Siendo éste el modelo más descargado de Hugging Face para la tarea de analizar la similitud entre dos oraciones. Este modelo pertenece a la familia de los *Sentence Transformers* desarrollado por (Reimers & Gurevych, 2019) y de igual manera que los otros dos modelos escogidos, este modelo es de código abierto y se puede descargar listo para el uso.

Finalmente, Pytesseract ha sido escogido como herramienta debido a su popularidad y a que es de código abierto.

A continuación, se detalla cada uno de los modelos seleccionados. Sin embargo, es importante describir previamente la arquitectura de un Transformer, ya que LayoutLMv2, legal-PEGASUS y all-MiniLM-L6-v2 están basados en esta arquitectura. Por la misma razón, se va a describir también la arquitectura del modelo BERT, que su vez, está basado también en la arquitectura de los *Transformers*. De esta manera, se irá explicando de manera incremental la arquitectura de cada modelo hasta llegar a los modelos utilizados en este TFG.

3.6.1 TRANSFORMER

Un *Transformer* es una arquitectura de redes neuronales presentada en 2017 por Google en el *artículo Attention is All You Need* de Vaswani, y otros, (2017). Esta nueva arquitectura pretende sustituir a los modelos recurrentes, como las LSTMs, en la tarea de generación de texto natural y solventar su problema de memoria cortoplacista. La principal propuesta de

esta nueva arquitectura gira en torno a un mecanismo de autoatención. A continuación, se resumen brevemente las características más importantes de esta nueva arquitectura, como son la representación de los datos de entrada en forma de *embeddings* y mecanismo de autoatención. Para una explicación más exhaustiva leer el artículo *Attention is All You Need*.

Arquitectura

El artículo propone un modelo basado en *encoder-decoder*, o codificador-decodificador. El *encoder* analizará el contexto de la frase de entrada y el *decoder* generará una nueva frase a partir de dicho contexto. El *encoder* está compuesto por 6 capas idénticas apiladas. Cada una de estas capas está a su vez compuesta por 2 subcapas. La primera de estas subcapas es un mecanismo de autoatención multi-cabeza. La segunda es una simple red neuronal seguida de una capa de normalización. El *decoder* también está compuesto por 6 capas idénticas apiladas. Además de las dos subcapas que incorpora el *encoder*, el *decoder* incorpora una tercera subcapa que aplica el mecanismo de autoatención sobre la salida del *encoder*. En la Figura 3.4 se puede observar la arquitectura de un *Transformer*, quedando claramente marcados el *encoder* y el *decoder*.

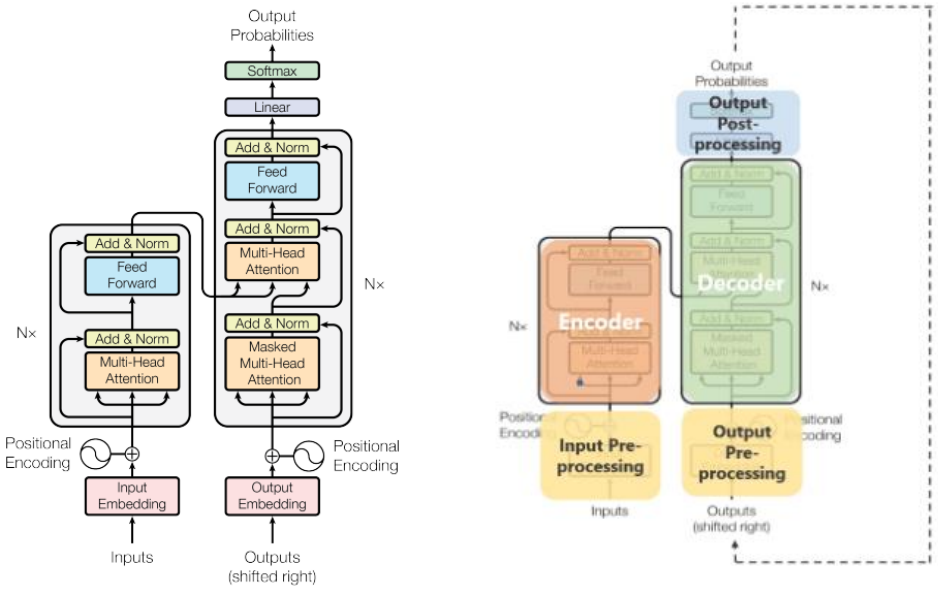

Figura 3.4. Diagrama de la arquitectura de un Transformer. Imagen extraída de (Vaswani, y otros, 2017).

Embeddings

Para que modelo sea capaz de trabajar con texto es necesario convertir este texto a un *embedding* de texto. Un *embedding* no es más que una representación vectorial de algo; texto o imágenes, por ejemplo. Para el caso específico de los *embeddings* de texto, existen diversos modelos que convierten texto a *embeddings*, como Word2Vec o WordPiece. Estos modelos tomarán como entrada un fragmento de texto, dividirán el texto en *tokens* y representarán cada *token* de manera vectorial. Estos modelos cuentan con un diccionario con miles de palabras o términos, denominados *tokens*. Al dividir el texto en *tokens*, lo que el modelo está haciendo es descomponer cada palabra en las palabras o términos que tenga guardados en su diccionario. Por ejemplo, el modelo descompondrá la palabra 'aeropuerto' en los *tokens* 'aero' + 'puerto'. Una vez el modelo ha descompuesto el texto en *tokens* devolverá una representación numérica de esta descomposición. Por otro lado, al tratar con secuencias de texto, es importante que el modelo tenga información de la posición de las palabras dentro de una frase para que pueda entender cómo se relacionan unas palabras con otras. Es por ello, que también se debe añadir un *embedding*, de la posición. Esto es simplemente un número indicando la posición de cada *token* de texto dentro de la oración.

Mecanismo de autoatención

El mecanismo de autoatención provee al modelo con una representación vectorial de la importancia de relación entre las palabras de una frase. Para ello, incorpora una estructura de varias cabezas de autoatención. Cada una de estas cabezas captará aspectos diferentes de las relaciones entre palabras, atribuyendo distintos pesos a cada relación según la relevancia de dicha relación, como puede observarse en la Figura 3.5. Los modelos basados en redes recurrentes dada su memoria cortoplacista, solo pueden referenciar cada palabra hasta un número limitado de palabras en el pasado. Este problema se solventa al introducir el mecanismo de autoatención, que permite a los *Transformers* utilizar el contexto completo de la frase y no solo una ventana determinada de texto como ocurría con los modelos recurrentes.

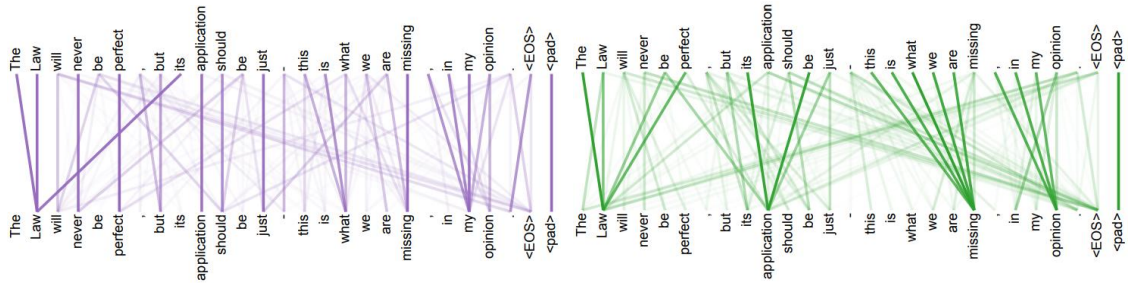

Figura 3.5. Vectores de atención generados por el mecanismo de autoatención.

Imagen extraída de (Vaswani, y otros, 2017).

Resultados

El modelo de *Transformers* obtiene mejores resultados que los anteriores modelos considerados el estado del arte, y sobre todo a un menor coste computacional. El modelo estableció una nueva puntuación máxima en BLEU de 28,4. Estos resultados confirman la eficacia del mecanismo de autoatención, no solo por los buenos resultados que obtiene el modelo, sino también por la eficiencia en cuanto a gasto computacional. Es por ello, que, a partir de la publicación de este modelo, son numerosos los modelos que han sido creado basados en este arquitectura, muchos de ellos mejorando los resultados del anterior estado del arte.

Model	BLEU		Training Cost (FLOPs)	
	EN-DE	EN-FR	EN-DE	EN-FR
ByteNet [18]	23.75			
Deep-Att + PosUnk [39]		39.2		$1.0 \cdot 10^{20}$
GNMT + RL [38]	24.6	39.92	$2.3 \cdot 10^{19}$	$1.4 \cdot 10^{20}$
ConvS2S [9]	25.16	40.46	$9.6 \cdot 10^{18}$	$1.5 \cdot 10^{20}$
MoE [32]	26.03	40.56	$2.0 \cdot 10^{19}$	$1.2 \cdot 10^{20}$
Deep-Att + PosUnk Ensemble [39]		40.4		$8.0 \cdot 10^{20}$
GNMT + RL Ensemble [38]	26.30	41.16	$1.8 \cdot 10^{20}$	$1.1 \cdot 10^{21}$
ConvS2S Ensemble [9]	26.36	**41.29**	$7.7 \cdot 10^{19}$	$1.2 \cdot 10^{21}$
Transformer (base model)	27.3	38.1	**$3.3 \cdot 10^{18}$**	
Transformer (big)	**28.4**	**41.8**	$2.3 \cdot 10^{19}$	

Tabla 3.1. Comparativa de resultados del modelo Transformer. Tabla extraída de (Vaswani, y otros, 2017).

3.6.2 BERT

BERT es un modelo considerado uno de los grandes modelos del lenguaje. Este modelo cuenta con una arquitectura multicapa de *Transformers* bidireccionales (Devlin, Chang, Lee, & Toutanova, 2019). Los desarrolladores de este modelo utilizan los *Transformers* como espina dorsal para crear un modelo capaz de realizar 11 tareas de Procesamiento de Lenguaje Natural. Este modelo es entrenado mediante el proceso de pre-entrenado más afinado. En este TFG se describe solamente el proceso de pre-entrenado, pero no el afinado, ya que las técnicas de pre-entrenado son comunes para todas las tareas finales y ayudarán a tener un mejor entendimiento del funcionamiento del modelo.

Embeddings

De manera similar a como ocurría con los *Transformers*, la entrada en forma de texto debe ser convertida a *embeddings*. En el caso de BERT, el texto de entrada serán una o dos frases. Esta vez, se representará cada frase como tres *embedding* distintos, como puede verse en Figura 3.6. En primer lugar, se crea un *embedding* de texto utilizando WordPiece. A este *embedding* de texto se le suma el mismo *embedding* de posición que encontrábamos en la arquitectura Transformer. Por último, BERT incorpora un tercer *embedding* que indica si cada *token* del texto pertenece a la primera o la segunda frase.

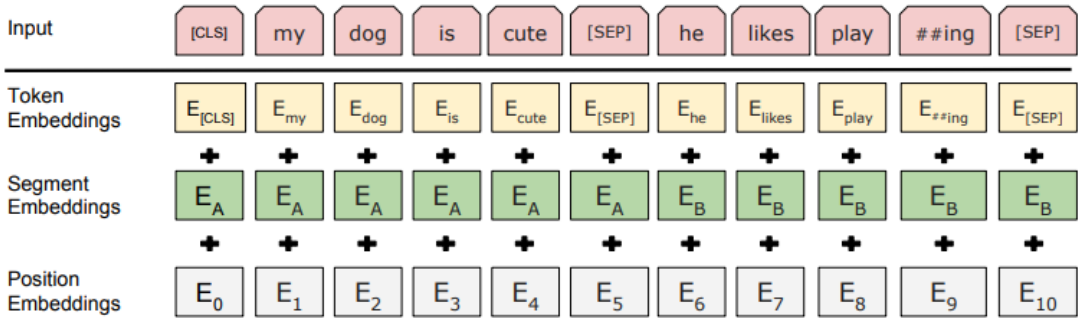

Figura 3.6. Embeddings de entrada de BERT. Imagen extraída de (Vaswani, y otros, 2017).

Pre-entrenado de BERT

Como se ha mencionado anteriormente, el modelo del lenguaje BERT se entrena en dos etapas. En primer lugar, el modelo es pre-entrenado con datos no etiquetados en tareas de aprendizaje autosupervisado y más tarde es afinado en tareas específicas con datos etiquetados. BERT es pre-entrenado para dos tareas simultáneamente. En la primera de estas tareas se enmascaran aleatoriamente un porcentaje de los tokens de entrada, que el modelo deberá predecir. Según el artículo de BERT, se eligió que un 15% de los token de entrada fueran enmascarados. De ese 15% de tokens enmascarados, un 80% fueron remplazados por el token [MASK], un 10% fueron remplazados por un token aleatorio y el 10% restante se dejaron intactos. La segunda tarea de pre-entrenamiento consiste en predecir si las dos frases dadas como datos de entrada están relacionadas o no. Esto ayuda al modelo a entender y detectar posibles relaciones entre dos frases. Para ello se añade al vector de datos de entrada una etiqueta *IsNext/NotNext* que nos indicará esto mismo. Para el entrenamiento un 50% de las frases estaban relacionadas y el 50% restante no tenían relación.

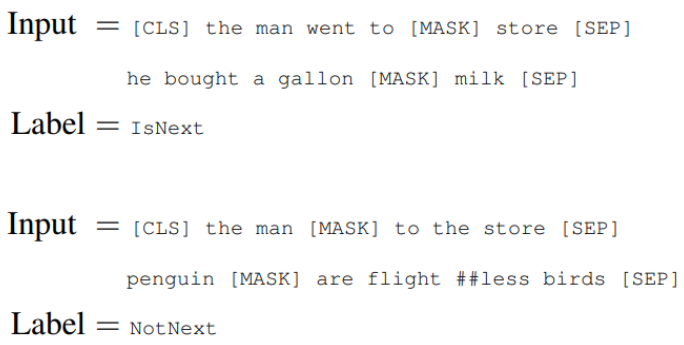

Figura 3.7. Datos de entrada de BERT. Imagen extraída de (Devlin, Chang, Lee, & Toutanova, 2019).

Como *datasets* para el pre-entrenamiento se utilizaron BooksCorpus, que cuenta con 800 millones de palabras, y Wikipedia en inglés, que cuenta con 2.500 millones de palabras. De Wikipedia se utilizaron solamente los fragmentos de texto, ignorando tablas, listas y encabezados.

Afinado de BERT

La explicación sobre el fine-tuneado de BERT queda fuera el alcance de esta breve explicación del modelo. Para una explicación detallada referirse al artículo original *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. Como breve resumen, cabe mencionar que BERT se afina en 11 tareas de NLP, como son la respuesta a preguntas o el resumen de texto. El desempeño del modelo en cada uno de los *benchmarks* utilizados y la comparación con otros famosos grandes modelos del lenguaje queda reflejado en la Tabla 3.2.

System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$BERT_{BASE}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$BERT_{LARGE}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Tabla 3.2. Resultados de BERT en distintas tareas de NLP. Tabla extraída de (Devlin, Chang, Lee, & Toutanova, 2019).

Resultados

Se puede observar que tanto las versiones *'Base'* y 'Large' de BERT tiene un rendimiento superior a modelos anteriores, como el famoso GPT de OpenAI. Es por esta razón, que su arquitectura, formada por *Transformers*, resulta un buen punto de partida para la construcción de un modelo que integre no solo información textual, sino también visual; como es el caso de la familia de modelos LayoutLM.

3.6.3 LAYOUTLM

Sobre todos los modelos desarrollados para resolver las tareas de *Document AI*, destaca la familia de modelos LayoutLM. Esta familia de modelos fue desarrollada por Microsoft y su primera versión del modelo salió a la luz en junio de 2020, con la publicación del artículo *LayoutLM: Pre-training of Text and Layout for Document Image Understanding.* Lo que propone este artículo es el análisis conjunto de texto y formato en un solo modelo.

Inspirándose en BERT, LayoutLM añade información visual y de posición 2-D sobre el texto encontrado en el documento. Es por ello, que para un correcto entendimiento de LayoutLM, se necesita entender el funcionamiento de BERT. No siendo LayoutLM el modelo que va a ser utilizado para el desarrollo de la herramienta propuesta en este TFG, es importante la compresión del funcionamiento de este modelo ya que LayoutLMv2 es una versión mejorada que comparte muchas características con LayoutLM.

Nuevos *Embeddings*

Como ya se ha mencionado, LayoutLM añade información visual a los datos de entrenamiento. Esto se consigue incorporando dos nuevos *embeddings* en los datos de entrada: (1) un *embedding* de posición 2-D, sobre la posición relativa del token en el documento, y (2) un *embedding* de imagen de la región donde se encuentra dicho texto. El *embedding* de posición 2-D se añade con el objetivo de dar información de posición que ayude al modelo a establecer relaciones entre los tokens. Esto se consigue estableciendo un origen de coordenadas en la esquina superior izquierda del documento y un recuadro de posición con coordenadas (x0, y0, x1, y1) para cada uno de los *tokens* de texto, donde (x0, y0) representan las coordenadas de la esquina superior izquierda del recuadro y (x1, y1) las de la esquina inferior derecha. Por otro lado, el *embedding* de imagen aporta información visual sobre el estilo y formato de cada *token*. Para generar estos *embeddings* de imagen cada región del documento donde se encuentra un *token* es procesada por un modelo Faster R-CNN (Ren, He, Girshick, & Sun, 2016), que extraerá las características de cada imagen y generará un serie de vectores que de manera numérica representan dichas características. A diferencia de como ocurría al analizar lenguaje natural, donde el texto ya viene dado, a la hora de analizar documentos se debe primero extraer el texto contenido en cada documento. Para ello, se puede debe emplear una herramienta de OCR externa, como PyTesseract. En la Figura 3.8, puede observarse que el embedding de posición 2-D se incluirá en el proceso de pre-entrenado, mientras que el embedding de imagen solo se utilizará para el afinado.

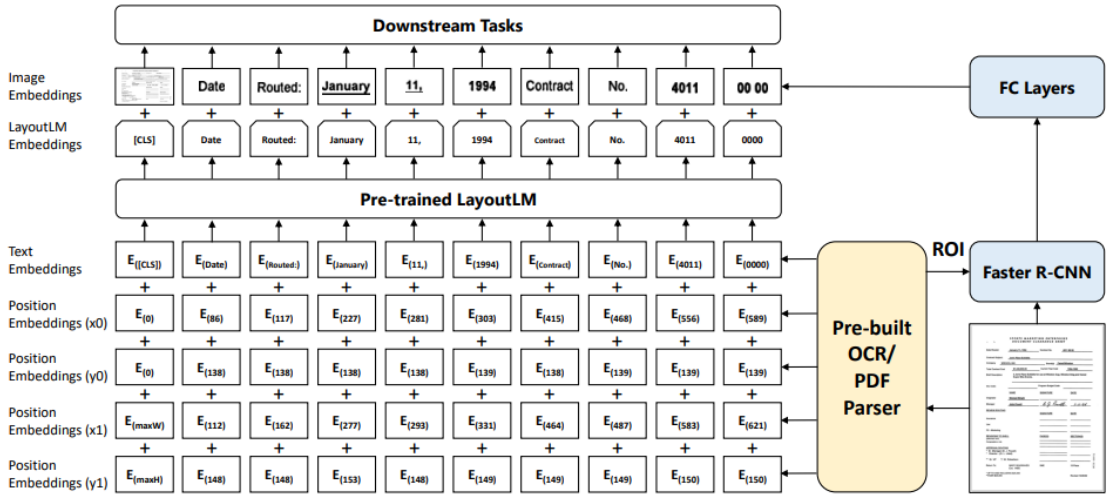

Figura 3.8. Arquitectura de LayoutLM e incorporación de los nuevos embeddings.

Imagen extraída de (Xu, y otros, 2020).

Pre-entrenado de LayoutLM

LayoutLM sigue un proceso de pre-entrenamiento en el que el modelo se entrena para dos tareas simultáneamente. En primer lugar, de manera similar a como se pre-entrenaba BERT, se enmascaran aleatoriamente un 15% de los tokens de entrada, aunque esta vez se mantiene su embedding de posición. De ese 15% de tokens seleccionados para enmascarar, un 80% se remplazarán por el token [MASK], otro 10% se remplazará por un token aleatorio, y el 10% restante se dejará intacto, tal y como sucedía con BERT. La segunda tarea de pre-entrenamiento consiste en clasificar el documento según su categoría. El objetivo de esta tarea es que el modelo aprenda a identificar características específicas de cada tipo de documento.

El rendimiento de los modelos pre-entrenado se ve muy condicionado al tamaño y calidad de los *datasets* utilizados, es por ello que su selección debe ser cuidadosa. Por esta razón, los autores escogieron la colección de documentos escaneados IIT-CDIP Test Collection, que cuenta con 11 millones de imágenes escaneadas. Cada imagen tiene adjunta un fichero XML con su texto y metadatos . El texto ha sido obtenido aplicando OCR a las imágenes,

mientras que los metadatos contienen información de las propiedades del documento, como el tipo de documento.

Afinado de LayoutLM

LayoutLM es afinado en tres tareas de específicas del entendimiento de documentos*: Form Undestanding*, *Receipt Understanding* y *Document Image Classificantion*. *Form Understanding* consiste en la extracción estructurada de texto contenido en formularios. Esta tarea se puede subdividir en dos subtareas: Etiquetado de *tokens* y la extracción de relaciones entre *tokens*. El etiquetado de *tokens* asigna a cada *token* una etiqueta de un conjunto de etiquetas predefinidas. La extracción de relaciones entre *tokens* se encarga de predecir posibles relaciones entre pares de *tokens*. LayoutLM se centra solamente en el etiquetado de *tokens*, asignado una etiqueta {*Beginning*, *Intermediate*, *End*, *Single*, *Other*} a cada uno de los *tokens*. En la tarea de *Receipt Understanding* el modelo deberá identificar y extraer una serie de campos determinados, como ya se explicó en la sección de 'Tareas del Document AI'. Por último, en la tarea de *Document Image Classification* el modelo deberá predecir el tipo de documento.

Los *datasets* utilizados para el afinado de cada una de estas tareas son los *benchmarks* estandarizados en el sector, como se ha mencionado anteriormente: FUNSD *dataset* para *Form Understanding*, SROIE para *Receipt Understanding* y RVL-CDIP para *Document Image Classification.*

Resultados

En la Tabla 3.3 se puede observar la comparativa entre los resultados obtenidos por LayoutLM en el *dataset* FUNSD y los obtenidos por dos modelos de Procesamiento del Lenguaje Natural consideramos el estado del arte en el momento de la publicación del artículo; estos son, el anteriormente mencionado, BERT y su versión optimizada, RoBERTa. Se puede observar como la incorporación de los nuevos *embeddings* de posición e imagen aumentan considerablemente los resultados obtenidos en este *dataset*, estableciendo LayoutLM un nuevo estado del arte con un F1 de 0,7927.

Modality	Model	Precision	Recall	F1	#Parameters
Text only	$BERT_{BASE}$	0.5469	0.671	0.6026	110M
	$RoBERTa_{BASE}$	0.6349	0.6975	0.6648	125M
	$BERT_{LARGE}$	0.6113	0.7085	0.6563	340M
	$RoBERTa_{LARGE}$	0.678	0.7391	0.7072	355M
Text + Layout MVLM	$LayoutLM_{BASE}$ (500K, 6 epochs)	0.665	0.7355	0.6985	113M
	$LayoutLM_{BASE}$ (1M, 6 epochs)	0.6909	0.7735	0.7299	113M
	$LayoutLM_{BASE}$ (2M, 6 epochs)	0.7377	0.782	0.7592	113M
	$LayoutLM_{BASE}$ (11M, 2 epochs)	0.7597	0.8155	0.7866	113M
Text + Layout MVLM+MDC	$LayoutLM_{BASE}$ (1M, 6 epochs)	0.7076	0.7695	0.7372	113M
	$LayoutLM_{BASE}$ (11M, 1 epoch)	0.7194	0.7780	0.7475	113M
Text + Layout MVLM	$LayoutLM_{LARGE}$ (1M, 6 epochs)	0.7171	0.805	0.7585	343M
	$LayoutLM_{LARGE}$ (11M, 1 epoch)	0.7536	0.806	0.7789	343M
Text + Layout + Image MVLM	$LayoutLM_{BASE}$ (1M, 6 epochs)	0.7101	0.7815	0.7441	160M
	$LayoutLM_{BASE}$ (11M, 2 epochs)	**0.7677**	**0.8195**	**0.7927**	160M

Tabla 3.3. Resultados de LayoutLM en FUNSD. Tabla extraída de (Xu, y otros, 2020).

3.6.4 LAYOUTLMV2

Con el objetivo de mejorar los resultados obtenidos con LayoutLM, apenas un año más tarde de la publicación del artículo que presentaba este modelo, se publica el siguiente modelo de la familia, LayoutLMv2 (Xu, y otros, 2021). Esta versión mejorada del anterior modelo propone cambios en la arquitectura del modelo y nuevas tareas de pre-entrenamiento. Los principales cambios se ven reflejados en los *embeddings* de entrada, un nuevo mecanismo de autoatención basado en posición 2-D y nuevas tareas y *datasets* de pre-entrenamiento.

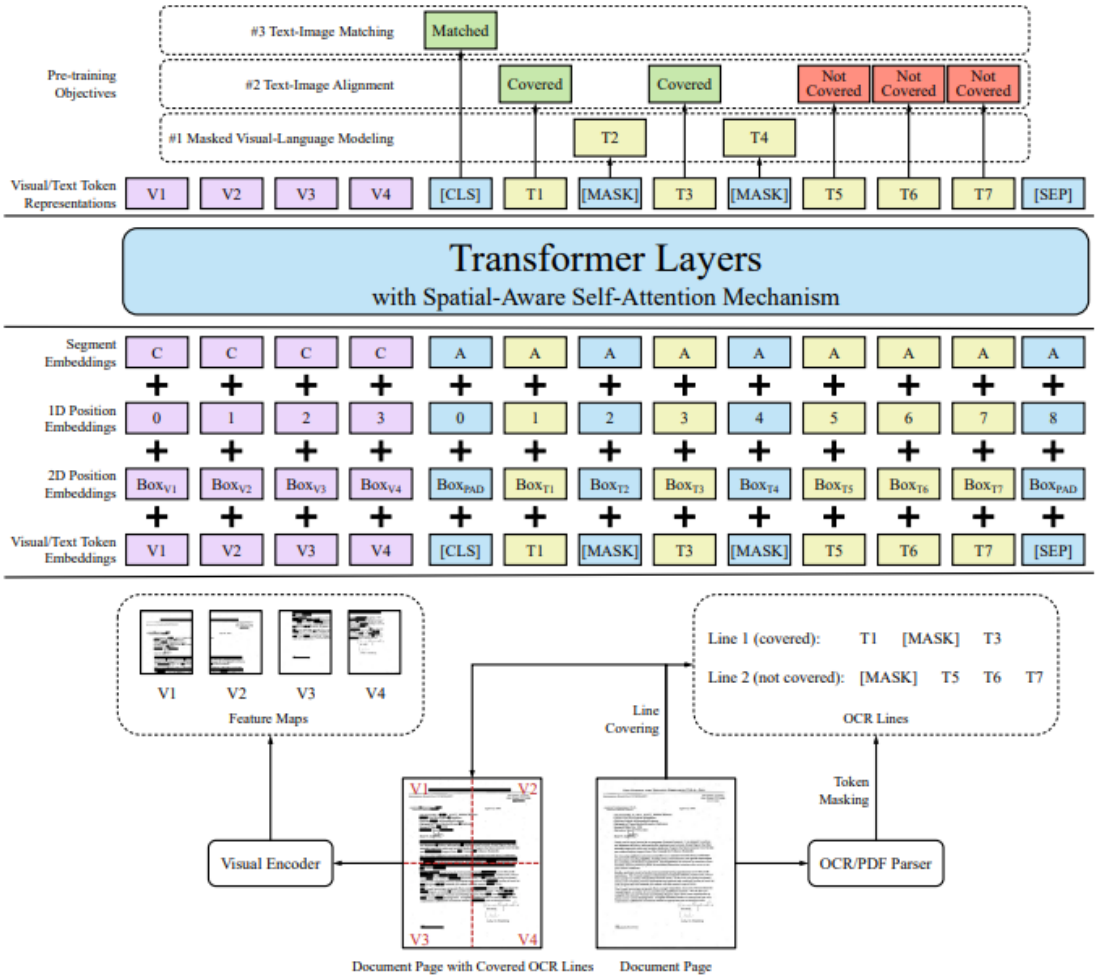

Figura 3.9. Arquitectura de LayoutLMv2 y representación de las tareas de pre-entrenamiento. Imagen extraída de (Xu, y otros, 2021).

Nuevos *Embeddings*

LayoutLMv2 utiliza también tres *embeddings* para representar el texto encontrado en el documento: un *embedding* de texto, generado por la combinación de una herramienta OCR integrada en el modelo y WordPiece, un *embedding* de posición 2-D, similar al utilizado en LayoutLM, y un *embedding* visual, que aporta información sobre el estilo y formato del texto. A diferencia de LayoutLM, donde los *embeddings* de imágenes solo se incorporan en el proceso de fine-tuneado, esta segunda versión incorpora información visual en el pre-entrenamiento también. Además de estos *embeddings* que representan el texto del

documento, se incorporan también otro *embedding* visual de la imagen del documento dividida en cuatro partes. Este nuevo *embedding* aportará información visual del conjunto del documento, y no solo de la región donde se encuentra cada *token*.

Mecanismo de autoatención basado en posición 2-D.

Basándose en el mecanismo de autoatención de los *Transformers*, los autores de LayoutLMv2 diseñan un nuevo mecanismo de un nuevo mecanismo de autoatención con conciencia espacial. El problema con el mecanismo de los *Transformers* es que solo tiene en cuenta la posición de absoluta 1-D de las secuencias de texto. Por ello, se crea y añade un nuevo mecanismo de autoatención basado en posición 2-D. De esta manera el modelo será capaz de aprender patrones locales que no varían de documento a documento, como que las oraciones se forman de derecha a izquierda o que las tablas tienen generalmente una estructura determinada.

Pre-entrenado de LayoutLMv2

De manera similar a LayoutLM, este nuevo modelo es entrenado en varias tareas de manera simultánea. La primera de estas tareas es común a LayoutLM, pero con algunas modificaciones. Se enmascaran de manera aleatoria un 15% de los *embeddings* de texto y visual, que el modelo deberá ser capaz de predecir. El *embedding* de posición se dejará intacto para que el modelo tenga información de la posición de los *embeddings* de texto y visual enmascarados. En el embedding visual de imagen de todo el documento las zonas donde se encuentren los *tokens* enmascarados también deberán ser enmascaradas, para que no se filtre información visual. En la segunda tarea de pre-entrenamiento, se cubren un 15% de los *tokens*. El modelo deberá decir para cada *token* si ha sido cubierto o no. La tercera y última tarea se basa en hacer al modelo predecir si el texto y las imágenes dadas al modelo pertenecen o no al mismo documento.
El *dataset* empleado para el pre-entrenamiento del modelo es ITT-CDIP, el mismo que se utilizó para LayoutLM.

Afinado de LayouLMv2

LayoutLMv2 se afina en tres tareas: *Visual Question Answering, Document Image Classification y Sequence Labelling*. Para afinar el modelo en la tarea de *Visual Question Answering* se utiliza el *dataset* DocVQA, para *Document Image Classification* el *dataset* RVL-CDIP y para *Sequence Labelling* se utilizan FUNSD, CORD, SROIE y el hasta ahora no mencionado, Kleister-NDA. Cabe remarcar que en la tarea de *Sequence Labelling* las etiquetas que se asignan a cada *token* son {'Other', 'Header-Beginning', 'Header-Intermediate', 'Question-Beginning', 'Question-Intermediate', 'Answer-Beginning', 'Answer-Intermediate'}. Estas etiquetas aportan información muy valiosa a la hora de formar relaciones entre *tokens.*

Resultados

Gracias a las mejoras incorporadas LayoutLMv2 consigue establecer un nuevo estado del arte en cada uno de los 6 *datasets* con los que ha sido afinado, mejorando también los resultados de su antecesor. Los resultados confirman la eficacia del nuevo mecanismo de autoatención basado en posición 2-D, las nuevas tareas de pre-entrenamiento propuestas e incorporar el *embedding* visual a la fase de pre-entrenamiento.

Model	Precision	Recall	F1	#Parameters
$\text{LayoutLM}_{\text{BASE}}$	0.7597	0.8155	0.7866	113M
$\text{LayoutLM}_{\text{LARGE}}$	0.7596	0.8219	0.7895	343M
$\text{LayoutLMv2}_{\text{BASE}}$	0.8029	0.8539	0.8276	200M
$\text{LayoutLMv2}_{\text{LARGE}}$	**0.8324**	**0.8519**	**0.8420**	426M
BROS (Anonymous, 2021)	0.8056	0.8188	0.8121	-

Tabla 3.4. Resultados de LayoutLMv2 en el dataset FUNSD. Tabla extraída de (Xu, y otros, 2021).

3.6.5 LEGAL PEGASUS

PEGASUS es un modelo del lenguaje basado en *Transformers* diseñado para la generación de secuencias de texto, en concreto para la tarea de resumir texto (Zhang, Zhao, Saleh, & Liu, 2020). Fue desarrollado por Google en 2020 y presentado por primera vez en la *2020 International Conference on Machine Learning*. El artículo presenta una nueva técnica de pre-entrenamiento autosupervisado llamada *gap-sentence generation.* En el momento de la publicación de este articulo el modelo consiguió resultados considerados el nuevo estado del arte en 12 *datasets* de resumen de texto.

Pre-entrenamiento con Gap-Sentence Generation

La hipótesis que plantea Google para justificar el desarrollo de esta nueva técnica es que cuanto más cerca este el objetivo del pre-entrenamiento de la tarea final del modelo mejor será el afinado del modelo. Esta nueva tarea de pre-entrenamiento consiste en enmascarar frases del texto de un documento y entrenar al modelo para que las prediga, como se puede observar en la Figura 3.10. Esta tarea tiene una dificultad enorme, por lo que los autores no esperan que el modelo prediga con precisión las frases enmascaradas. Sin embargo, esta tarea fuerza al modelo a aprender sobre el propio lenguaje.

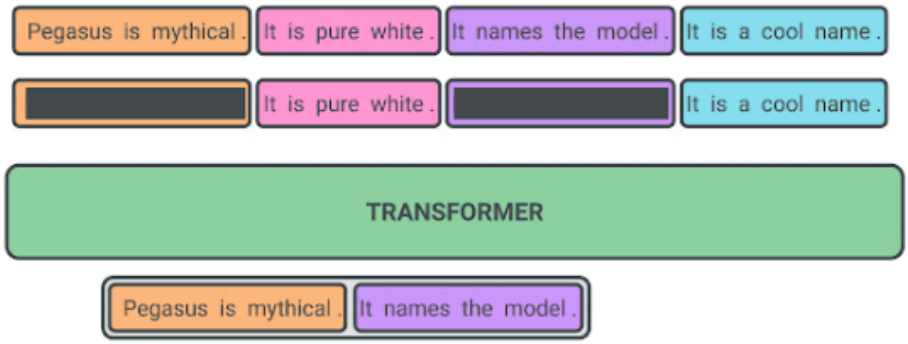

Figura 3.10. Ejemplo de entrada de pre-entrenamiento en PEGASUS. Imagen extraída de https://ai.googleblog.com/2020/06/pegasus-state-of-art-model-for.html.

Como *dataset* de pre-entrenamiento, los autores escogieron C4 y HugeNews. C4 es un *dataset* compuesto por 350 millones de páginas web, mientras que HugeNews cuenta con 1,5 mil millones de artículos variados como artículos periodísticos, artículos científicos, patentes o documentos legales.

Afinado

La versión de PEGASUS utilizada en este TFG fue afinada en primer lugar en el *dataset* CNN/DailyMail, estableciendo un nuevo estado del arte. Como se ha mencionado anteriormente, para obtener mejores resultados, es importante que el *dataset* con el que se afina el modelo sea lo más parecido al uso real que se le quiera dar al modelo. Es por ello, que una vez afinado el modelo con CNN/DailyMail, se ha buscado una versión de PEGASUS afinada con documentos legales. En HuggingFace (https://huggingface.co/nsi319/legal-pegasus) se puede encontrar una versión del modelo afinada en una colección de más de 2700 comunicados y reclamaciones de litigios de la SEC (*U.S. Securities and Exchange Commision*). La autoría de este modelo afinado es del usuario *nsi319* en HuggingFace.

3.6.6 ALL-MINILM-L6-V2

All-MiniLM-L6-v2 es el modelo para la tarea de *Sentence Similarity*, o encontrar la similitud entre dos oraciones, más descargado de HuggingFace. Es una versión del modelo *MiniLM* desarrollado por Microsoft (Wang, y otros, 2020), que pertenece a la familia de los *Sentence Transformers* (Reimers & Gurevych, 2019). Esta familia de modelos es entrenada con el objetivo de hallar la similitud entre dos oraciones. Para ello, los *datasets* de entrenamiento están formados por pares de oraciones y su respectiva puntuación de similitud. El *encoder* de estos modelos calcula los *embeddings* de estos pares de frases y posteriormente se calcula la similitud de estos *embeddings* con el algoritmo de similitud del coseno. Esta similitud calculada es comparada con la similitud real anotada en el *dataset*, penalizando al modelo por el error cometido. Recordemos que los *embeddings* generados por el *encoder* de un *Transformer* son representaciones numéricas del texto y de las relaciones generadas entre palabras por el mecanismo de autoatención. De esta manera el modelo aprenderá a generar

embeddings que al ser procesados con el algoritmo de similitud del coseno informen con certitud de la similitud entre dos oraciones.

3.6.7 PYTESSERACT OCR

Python-tesseract, o Pytesseract, es una herramienta de Python de reconocimiento óptico de caracteres. Esta herramienta es un envoltorio, o *wrapper*, de la herramienta de Google Tesseract-OCR. Este OCR fue inicialmente desarrollado por Hewlett-Packard entre 1985 y 1994. En 2005 la empresa hizo el proyecto de código abierto. Meses más tarde, en noviembre de 2006, Google continuó con el desarrollo de esta herramienta, hasta 2018. A día de hoy, Tesseract-OCR sigue siendo una herramienta de código abierto y Google se encarga de su mantenimiento. Es por ello es la opción preferida de muchos desarrolladores a la hora de elegir un OCR.

El reconocimiento óptico de caracteres es el proceso por el cual se detectará y leerá texto de imágenes, ya sea éste manuscrito o digital. Este proceso se divide generalmente en una serie de subprocesos (Sakira, Sirisala, & Velpuru, 2021), como puede verse en la Figura 3.11:

- El primer paso es el preprocesamiento de las imágenes. Es común que el origen de las imágenes no sea digital, por lo que éstas pueden contener destellos, sombras, ruido o estar borrosas. Es por esta razón, que es necesario aplicar un preprocesado para facilitar la extracción de las imágenes. Algunas técnicas de preprocesado son:
    - Enderezar, inclinar, rotar y recortar la imagen.
    - Convertir la imagen a una escala de grises o incluso a blanco y negro.
    - Eliminar ruido y manchas digitales.
    - Escalar el tamaño de la imagen al tamaño admitido por el procesador OCR.

- Una vez el documento ha sido procesado la herramienta debe reconocer las zonas donde hay texto y segmentarlas. En este subproceso el texto es generalmente segmentado en párrafos, que posteriormente son divididos en palabras y finalmente en caracteres individuales.
- A continuación, las características principales de cada carácter encontrado en la imagen son extraídas.

- Una vez se han extraído las características de cada carácter, éste será clasificado como un carácter ASCII. Una vez se han clasificado todos los caracteres, la herramienta vuelve a formar las palabras.

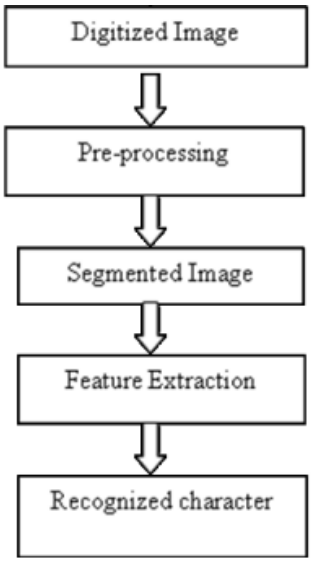

Figura 3.11. Proceso de extracción de texto por un OCR. Imagen extraída de (Sakira, Sirisala, & Velpuru, 2021).

Capítulo 4. HERRAMIENTA DESARROLLADA

4.1 ARQUITECTURA

Una vez descritos los modelos que van a ser utilizados en la herramienta desarrollada en este TFG, se va a detallar la arquitectura de la herramienta y cómo cada uno de estos modelos ayuda a la consecución de los objetivos propuestos. En la Figura 4.1 se muestra la arquitectura de la herramienta. Por otro lado, en el Anexo II se puede ver imágenes completas del análisis paso a paso de dos pólizas de seguro distintas.

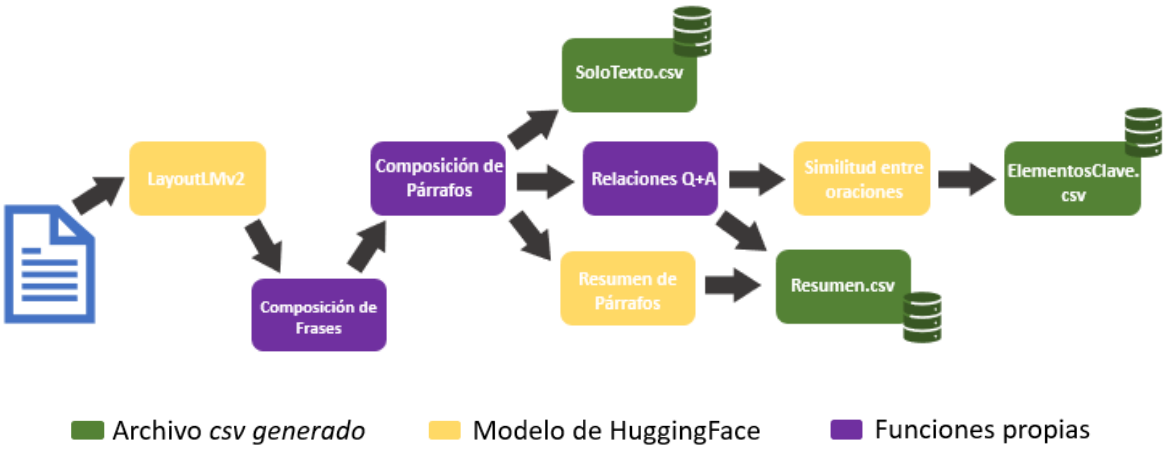

Figura 4.1. Esquema de la arquitectura de la herramienta desarrollada en este TFG.

4.1.1 DOCUMENTOS ADMITIDOS

La herramienta admitirá solamente documentos en formato de imagen de una sola página. La implementación de una función que permita analizar documentos multi-página se deja como posible trabajo futuro.

4.1.2 PROCESADO CON LAYOUTLMV2

El primer paso en el procesado del documento es procesarlo con el modelo LayoutLMv2. Como ya se ha detallado, este modelo etiquetará cada uno de los tokens encontrados en el

documento con una de las etiquetas {'Other', 'Header-Beginning', 'Header-Intermediate', 'Question-Beginning', 'Question-Intermediate', 'Answer-Beginning', 'Answer-Intermediate'}. En la Figura 4.2 se muestra un ejemplo del etiquetado de *tokens* por LayoutLMv2. Estas etiquetas aportan información muy útil sobre cada palabra, ya que nos informan del tipo de palabra y de posibles relaciones entre palabras. Sin embargo, existen dos inconvenientes que nos impiden la consecución de los objetivos propuestos utilizando solamente esta información:

- El modelo etiqueta *tokens* y no frases. La información que aporta un solo *token* aislado es escasa, pudiendo llegar a ser nula, ya que el significado de una sola palabra, sin el contexto de su frase, pierde por completo su valor informativo. Es por ello que resulta primordial la formación de frases y párrafos a partir de los *tokens.* Esta tarea se ve facilitada por el hecho de que las etiquetas contienen información sobre la posición de cada palabra dentro de una frase (*Beginning* o *Intermediate*).
- LayoutLMv2 etiqueta cada *token*, pero no crea relaciones entre cada uno de ellos. Estas relaciones son necesarias si se quieren extraer par clave-valor, o simplemente si se quiere extraer la información contenida en el documento de manera estructurada. Por este motivo, la búsqueda y extracción de relaciones entre *tokens* es esencial. Las etiquetas de tipo 'Question' y 'Answer' ayuda en el proceso de establecer dichas relaciones.

Un posible problema a la hora de analizar un documento con LayoutLMv2 es que el documento contenga más elementos de los que pueden ser analizados por el modelo, debido a su tamaño. Como se indica en la guía de 'Cómo utilizar la herramienta' presente en el propio cuaderno de Colab, en caso de darse este error, la imagen deberá ser recortada. Se dejará como trabajo futuro el desarrollo de una función que solvente este error.

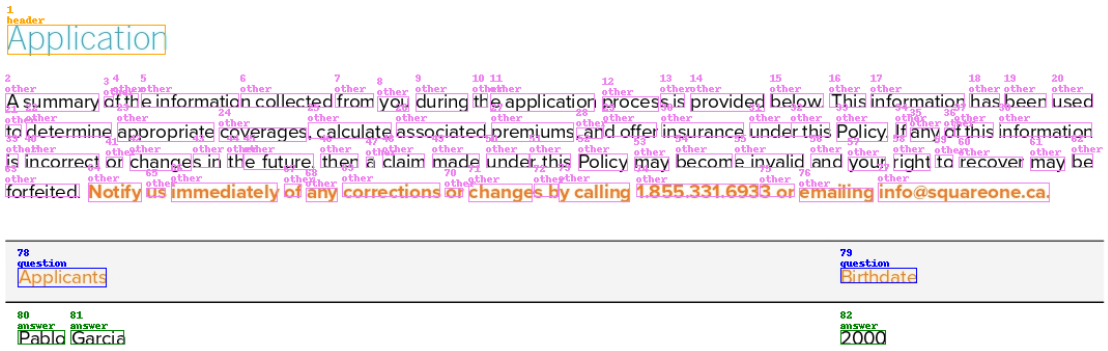

Figura 4.2. Ejemplo de procesado de un póliza de seguro con LayoutLMv2.

4.1.3 COMPOSICIÓN DE FRASES DE UNA LÍNEA

Como se acaba de mencionar, la composición de frases a partir de *tokens* etiquetados es esencial. Para ello se ha creado una función que tomando información de las etiquetas asignadas a cada *token* por LayoutLMv2 y su posición relativa decidirá si cada *token* y el de su derecha pertenecen o no a la misma frase. Los criterios para decidir si dos *tokens* pertenecen a la misma frase serán los siguientes: si un *token* es de un tipo, siendo los tipos {'Other', 'Header', 'Question', 'Answer'} y el de su derecha es del mismo tipo, ambos *tokens* pertenecerán a la misma frase siempre y cuando el *token* de la derecha no haya sido previamente asignado a otra frase. Esta función tendrá información de cuál es el *token* que cada *token* tiene a la derecha gracias a las funciones *nextRight* y *nextBelow*. Una vez se ha iterado por todos los *tokens* del documentos se crearán nuevos recuadros conteniendo las frases formadas, tal y como se muestra en la Figura 4.4. Cada una de estas frases será del mismo tipo que el primer *token* que la compone.

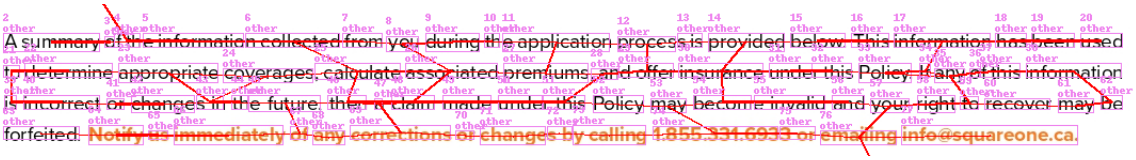

Figura 4.3. Texto mostrando el token más próximo hacia la derecha y hacia abajo.

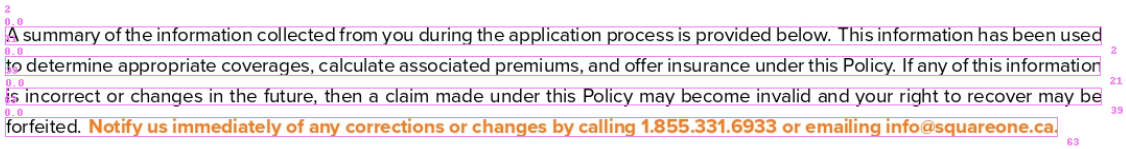

Figura 4.4. Texto con frases formadas.

Funciones *nextRight* y *nextBelow*

En el lenguaje escrito las palabras y frases se forman de izquierda a derecha y de arriba abajo. Por este motivo, se han creado las funciones *nextRight* y *nextBelow*. Estas dos funciones encontrarán para cada elemento del documento el elemento más cercano hacia la derecha y hacia abajo. Para ello, simplemente se calculará la distancia de cada elemento a cada otro y se hallará la mínima de todas estas distancias. Estos elementos pueden ser tanto *tokens*, como palabras, frases o párrafos. Las funciones *next* tomarán como entrada las coordenadas de cada elemento en el documento y unos valores máximos de distancia vertical y horizontal. Estos valores máximos establecen el límite de distancia vertical hasta la que la función *nextRight* buscará el *token* más próximo y el límite de distancia horizontal hasta la que la función *nextBelow* buscará el *token* más cercano. Estos límites se establecen ya que, por ejemplo, puede darse el caso de que el *token* a menos distancia en el eje horizontal esté a mucha distancia en el eje vertical, generalmente, estos dos *tokens* no estarán relacionados ya que solo están a poca distancia en eje horizontal, pero no en el vertical. A lo largo del desarrollo de la herramienta de este TFG se utilizan pequeñas variaciones de estas funciones.

```
def nextRight(x_center, y_center, width_range_factor, height_range_factor):

def nextBelow(x_center, y_center, width_range_factor, height_range_factor):
```

Figura 4.5. Definición de las funciones nextRight y nextBelow.

4.1.4 COMPOSICIÓN DE PÁRRAFOS

De manera similar a como se han compuesto frases a partir de *tokens*, el próximo paso es componer párrafos a partir de frases de una línea. Para ello se van a utilizar de nuevo las funciones *nextRight* y *nextBelow*, esta vez proveyéndolas con las coordenadas de las frases.

La lógica para la composición de los párrafos se basa de nuevo en unir en un mismo párrafo las frases contiguas que sean del mismo tipo. Esta vez se unirán frases que estén una debajo de la otra, y no a la derecha. Ocurrirá en numerosos casos que una frase no sea unida a ninguna otra frase; en esos casos los párrafos estarán compuestos por una sola frase.

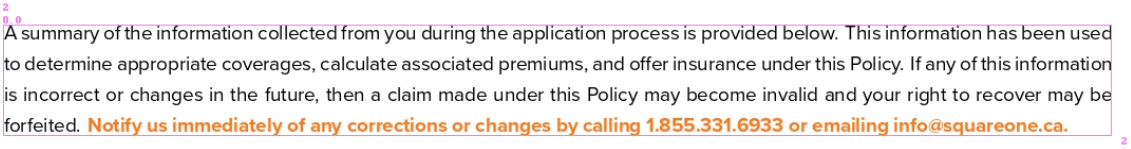
A summary of the information collected from you during the application process is provided below. This information has been used to determine appropriate coverages, calculate associated premiums, and offer insurance under this Policy. If any of this information is incorrect or changes in the future, then a claim made under this Policy may become invalid and your right to recover may be forfeited. Notify us immediately of any corrections or changes by calling 1.855.331.6933 or emailing info@squareone.ca.

Figura 4.6. Párrafo de texto formado.

Una vez se han formado todos los párrafos, se van a realizar tres tareas. En primer lugar, se va a extraer todo el texto contenido en estos párrafos. De esta manera se cumplirá el objetivo de extraer todo el texto presente en el documento. En segundo lugar, se van a buscar relaciones para unir párrafos de tipo 'Question' con sus 'Answer' correspondientes. Por último, se van a procesar los párrafos de tipo 'Other' con el objetivo de resumirlos sin perder información relevante.

4.1.5 SOLOTEXTO.CSV

El primer objetivo de esta herramienta es la extracción de todo el texto contenido en el documento. Para ello, se va a utilizar la herramienta OCR Pytesseract. A esta herramienta se le proveen una a una con las regiones de la imagen del documento donde se encuentra cada párrafo. Pytesseract extrae el texto de cada una de estas imágenes, que es guardado en un fichero *csv* llamado *SoloTexto.csv*.

4.1.6 BÚSQUEDA DE RELACIONES 'QUESTION' + 'ANSWER'

El objetivo de esta función es encontrar y formar relaciones entre preguntas y respuestas, de tal manera que cada pregunta tenga asociada una o dos respuestas. Para ello se va a asumir que todas las formaciones de 'Pregunta + Respuestas' siguen alguna de las siguientes seis estructuras representadas en la Figura 4.7. Estas seis estructuras son todas las combinaciones posibles asumiendo que las respuestas siempre van a estar a la derecha o debajo de las preguntas. Estas asunciones están basadas principalmente en observaciones, pero también se fundamentan en la construcción del lenguaje escrito, donde, como ya se ha mencionado con anterioridad, las frases de construyen de derecha a izquierda y de arriba abajo.

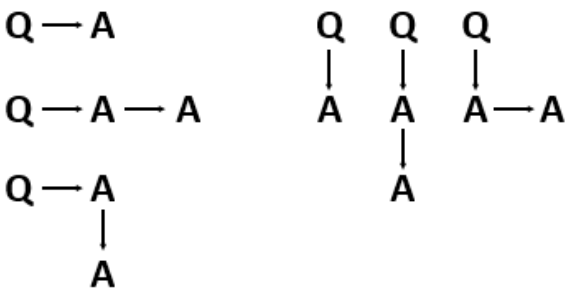

Figura 4.7. Combinaciones posibles de relaciones 'Pregunta + Respuestas'.

Una vez más se han utilizado las funciones *nextRight* y *nextBelow* para encontrar los párrafos contiguos. Tras ello, se han creado dos procesos de iteración. En el primero, simplemente se ha iterado por todos los párrafos comprobando si una párrafo es del tipo 'Question' y su contiguo, ya sea hacia la derecha o hacia abajo, es del tipo 'Answer'. En caso afirmativo se guardarán los índices de estos pares 'Pregunta + Respuestas' en un vector. La segunda iteración itera solamente por este recién creado vector de pares 'Pregunta + Respuestas', y comprueba si párrafo contiguo, hacia la derecha o hacia abajo, de cada párrafo del tipo 'Answer' guardado en el vector es otra párrafo del tipo 'Answer'. En caso afirmativo, se añadirá el índice de este párrafo al vector anterior. De esta manera, al finalizar estos procesos iterativos, se habrá generado un matriz de tres columnas con el índice la pregunta en la primera columna y los índices de una o dos respuestas en las columnas 2 y 3. El número de filas dependerá del número de relaciones que encuentre el algoritmo.

4.1.7 RESUMEN DE PÁRRAFOS

En alineación con los objetivos de este Trabajo de Fin de Grado, a la hora de extraer información contenida en párrafos es necesario resumir dichos párrafos para que su lectura sea más fácil e inmediata. Para ello se va a utilizar el ya mencionado legal-PEGASUS. De la totalidad del texto encontrado en el documento se va a resumir solamente aquellos párrafos etiquetados como 'Other' que tengan una extensión determinada que podrá ser establecida por el usuario.

Para establecer la longitud máxima y mínima del resumen generado por el modelo se han elegido las funciones:

$$\text{longitud máxima del resumen} = \sqrt{N * longitud\ del\ texto}$$

$$\text{longitud mínima del resumen} = \sqrt{M * longitud\ del\ texto}$$

Estas longitudes máxima y mínimas serán dadas como parámetro al legal-PEGASUS. Se ha encontrado satisfactorio establecer N = 10 y M = 30. De esta manera los texto cortos no se resumirán y los de mayor extensión se resumirán en mayor medida.

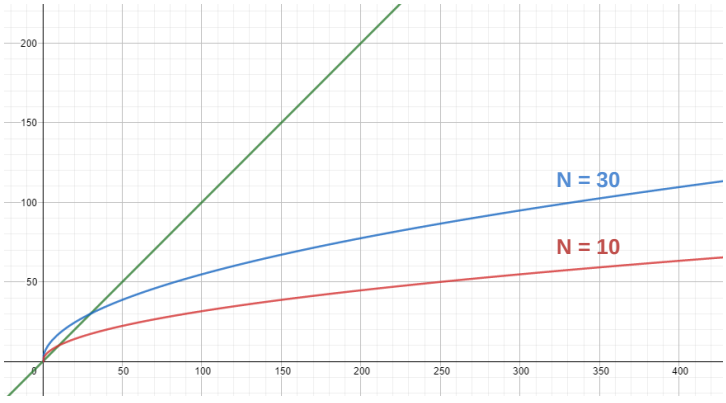

Figura 4.8. Gráfica de la longitud máxima y mínima del resumen del texto para N=10 y M=30.

4.1.8 RESUMEN.CSV

Una vez se han encontrado todos lo pares de 'Pregunta + Respuestas' y se han resumido todos los párrafos de tipo 'Other' se extrae toda esta información con Pyteseract y se guarda

en un archivo llamado *Resumen.csv*. Este archivo servirá a modo de resumen de toda la información contenida en el documento. A diferencia del archivo *SoloTexto.csv*, en este archivo encontraremos la información de manera más estructurada gracias a los pares de 'Pregunta + Respuestas'.

4.1.9 SIMILITUD ENTRE ORACIONES

Como paso final en el procesamiento de la información, se van a analizar las similitudes entre todas las preguntas de los pares 'Pregunta + Respuestas' y una serie de diccionarios que contienen elementos que se pretenden encontrar en el documento. Estos diccionarios han sido creados tomado como referencia las listas de elementos comunes encontrados en las pólizas de seguros, que se detallan en la sección de 'Estructura de una póliza de seguros'. La herramienta tiene tres diccionarios de manera predeterminada. Sin embargo, la creación de nuevos diccionarios personalizados es posible, y en la guía de 'Cómo utilizar la herramienta' se explica cómo hacerlo. Los diccionarios predeterminados son:

- Diccionario genérico a todas las pólizas de seguros.
- Diccionario específico para seguros del automóvil.
- Diccionario específico para seguros del hogar.

La similitud ente preguntas y diccionarios se haya dándole al modelo all-MiniLM-L6-V2 una pregunta y un elemento del diccionario para que halle la similitud entre ellos. El modelo calificará la similitud entre cada par con valores del 0 al 1. Se itera hasta que se haya hallado la similitud de cada pregunta con cada elemento de cada diccionario. Una vez hecho esto, se selecciona para cada elemento de los diccionarios la pregunta que mayor similitud tenga con él. Si esta similitud es mayor que un cierto umbral establecido por el usuario, se extrae este par. Si no existe una similitud mayor o igual al umbral establecido por el usuario, se devuelve el mensaje 'N/A'.

4.1.10 ELEMENTOSCLAVE.CSV

Todos estos pares se guardan en un archivo *csv* llamado *ElementosClave.csv* que contendrá, si se han encontrado, todos los elementos clave que se deseaban extraer del documento.

Capítulo 5. ANÁLISIS DE RESULTADOS

Una vez desarrollada la herramienta de extracción de datos, se procede al análisis de los resultados obtenidos al procesar distintos documentos y a la comparativa de estos resultados con los ofrecidos por herramientas ya existentes, como Document AI de Google. Los documentos utilizados para el análisis de resultados son pólizas de seguros extraídas de internet. En concreto, se van a analizar una póliza de seguro del automóvil y otra de seguro del hogar. Estas pólizas de seguros en ningún momento han sido modificadas o manipuladas con el objetivo de obtener mejores resultados.

5.1 RESULTADOS OBTENIDOS CON DOCUMENT AI DE GOOGLE

En primer lugar, se van a analizar los resultados ofrecidos por la herramienta de Google Document AI en las tareas de extracción de todo el texto de un documento, extracción de relaciones y extracción de elementos clave. Se ha utilizado la plantilla 'General', ya que, como se comentó anteriormente, Google no dispone de una plantilla específica para pólizas de seguro, y la creación de un plantilla personalizada requiere de al menos 200 pólizas de seguros etiquetadas.

Extracción de todo el texto

La extracción de texto con la herramienta Document AI es muy precisa. En ambas pólizas de seguro la herramienta de Google es capaz de extraer todo el texto presente en el documento si ningún tipo de error, tal y como se observa en la Figura 5.1.

Figura 5.1. Ejemplos de extracción de texto con OCR por la herramienta Document AI de Google.

Extracción de relaciones

En la tarea de extracción de relaciones la herramienta de Google es capaz de establecer algunas relaciones entre los distintos elementos del documento. De la póliza del seguro del automóvil, la herramienta fue capaz de formar un total de 26 relaciones. Sin embargo, estas relaciones a menudo no tienen sentido, como se muestra en la Figura 5.2. Por ello, se considera que de las 26 relaciones extraídas solo 18 de ellas tienen sentido. En cuanto a la póliza del hogar, la herramienta de Google formó 8 relaciones, de las que 2 no se consideran válidas.

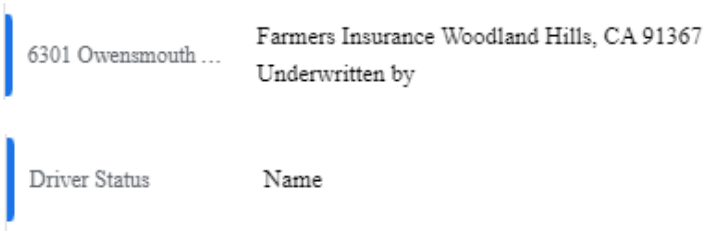

Figura 5.2. Ejemplos de relaciones no válidas.

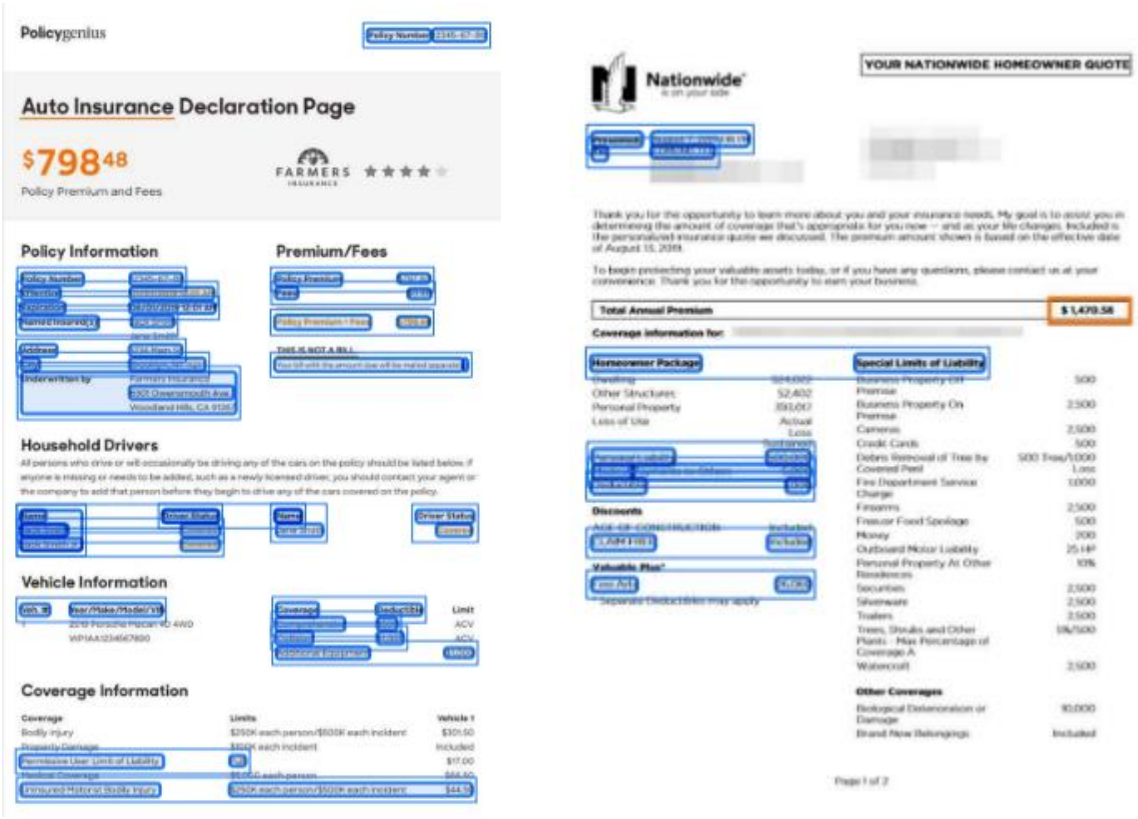

Figura 5.3. Ejemplos de extracción de relaciones por la herramienta Document AI de Google.

Extracción de tablas

Document AI también extrae las tablas que encuentra en el documento. Se puede observar en la Figura 5.4 que los resultados de esta tarea varían mucho de una póliza a otra. En la póliza de seguro del automóvil las tablas formadas son bastante fieles a las encontradas

realmente en el documento, aunque alguna tabla no ha sido encontrada. Sin embargo, en la póliza de seguro del hogar las tablas formadas no son precisas.

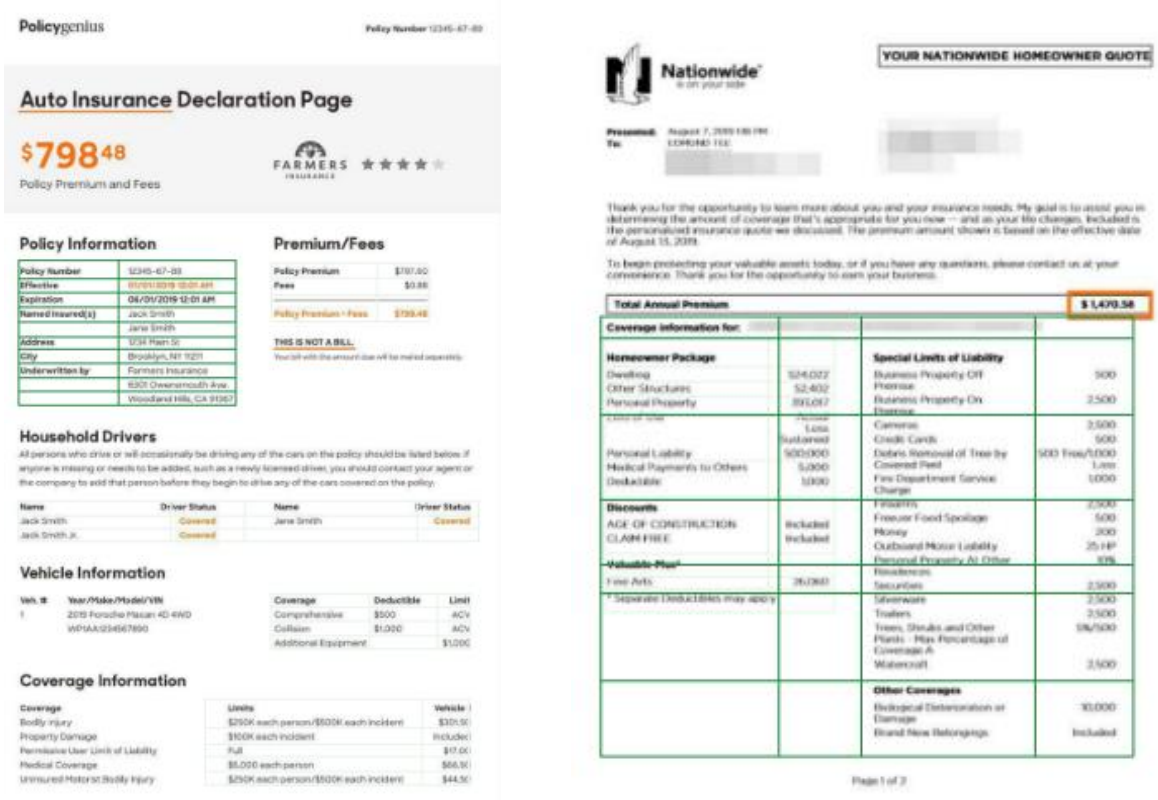

Figura 5.4. Ejemplos de formación de tablas por la herramienta Document AI de Google.

Extracción de elementos clave

El uso la platilla de tipo ‘General’ de Document AI no permite la extracción de elementos clave. Para poder hacerlo sería necesario crear un plantilla personalizada, debiéndole proveer de un mínimo de 200 documentos anotados con los campos que se desean extraer.

5.2 RESULTADOS OBTENIDOS CON LA HERRAMIENTA DESARROLLADA EN ESTE TRABAJO DE FIN DE GRADO

Todos los resultados mencionados en esta sección pueden ser encontrados en el repositorio de GitHub del proyecto, en la carpeta de 'Resultados'. En el Anexo III se adjunta un enlace a dicho repositorio.

Extracción de texto con OCR

El primer archivo *csv* que genera la herramienta, 'SoloTexto.csv', es un archivo que contiene todo el texto encontrado en el documento. En la póliza de seguro del automóvil la herramienta encuentra un total de 82 fragmentos de texto a extraer. Estos fragmentos de texto son los llamados 'Párrafos' en la descripción de la herramienta, que pueden verse marcados según su tipo en la Figura 5.5. De estos 82 párrafos: 71 han sido extraídos completamente bien, 4 contienen fallos parciales o no están completos y 7 han sido extraídos erróneamente. Esto supone una tasa de acierto de alrededor del 89%. En la póliza de seguro del hogar la herramienta detecta 77 párrafos a extraer, de los cuales 24 son extraídos erróneamente, 13 con fallos parciales, dejando 40 elementos bien extraídos. Esto hace que la herramienta tenga una tasa de acierto del 60%.

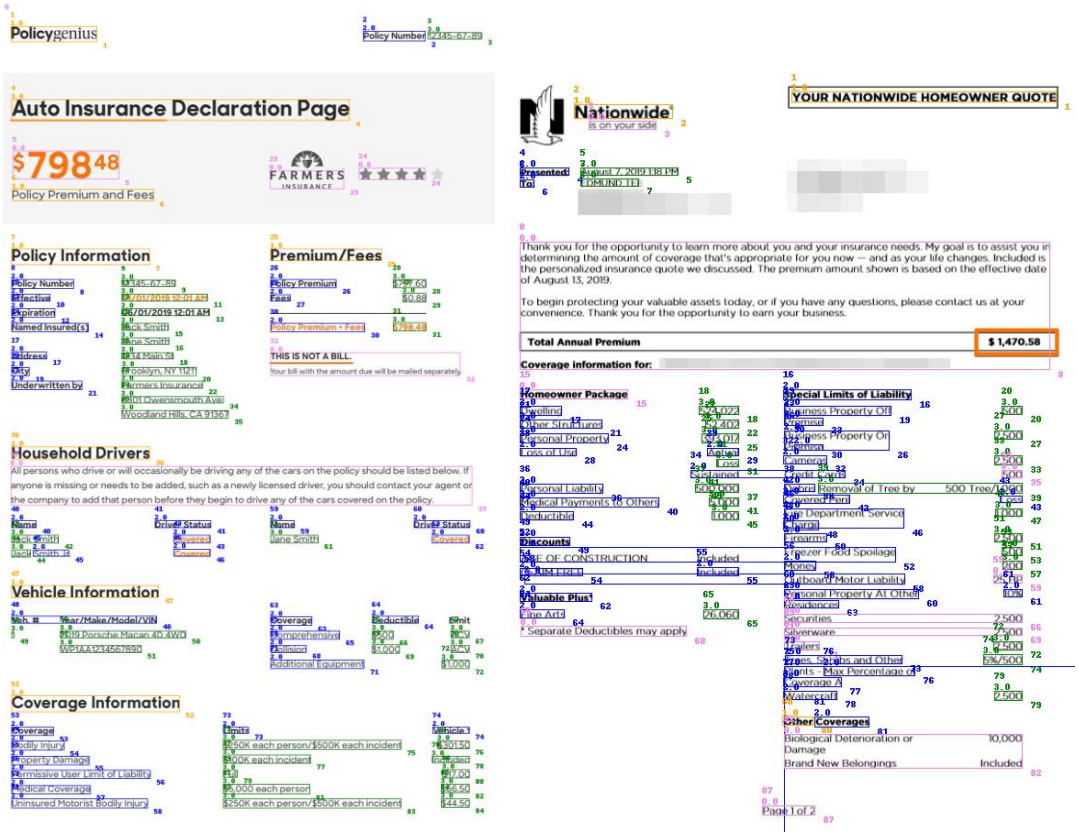

Figura 5.5. Regiones seleccionadas para la extracción de texto con OCR.

Extracción de relaciones

En el archivo 'Resumen.csv' se puede encontrar las relaciones de pares 'Pregunta + Respuestas' creados por la herramienta desarrollada en este TFG. Esta herramienta es capaz de extraer un total de 27 relaciones de preguntas y respuestas de la póliza de seguro del automóvil. De estas relaciones, se considera que 25 de ellas son relaciones con sentido que aportan cierta información. Sin embargo, la información contenida en 4 de estas relaciones está ausente debido a que la herramienta no ha extraído correctamente el texto. En la póliza de seguro del hogar la herramienta forma 26 relaciones. De estas 26 relaciones la información de 16 de ellas no está completa por la falta de precisión en la extracción de texto, dejando solo 10 relaciones con sentido.

PREGUNTA	RESPUESTA 1	RESPUESTA 2
Policy number	2345-67-89	
Pollcy number	12345~67-89	
Effective	(01/04/2019 12:07 AM.	
Expiration	06/01/2019 12:01AM	
Named insured(s)	Jack Smith	Jane Smith
Address	1234 Main St	
	Brooklyn, NY 11211	
Underwritten by	Farmers Insurance	6301 Owensmouth Ave.
Policy premium	$797.60	
	$0.88	
Policy premium + fees:	$798.48	
	Jack Smith.	
Covered	Jane Smith	
Jack	Smith Jr.	
Veh. # year/make/model/vin	2019 Porsche Macan 4D 4wD-	
Bodily injury	$250K each person/$500K each incident	
Property damage	$100K each incident	Included
Permissive user limit of liability	Full	$17.00
Medical coverage	$5,000 each person	$66.50
Uninsured motorist bodily injury	$250K each person/$500K each incident	$44.50
Name	Jane Smith	
Deductible	3500	ACV
Comprehensive	3500	ACV
Collision	$1,000	ACY
Additional equipment:	$1,000	
Limits	$250K each person/$500K each incident	
Vehicle t	$301.50	Included

Tabla 5.1. Resumen.csv generado por la herramienta.

Extracción de elementos clave

Por último, en el archivo 'ElementosClave.csv' encontramos un lista con los elementos clave encontrados y extraídos de ambas pólizas.

Los elemento clave objetivo están contenidos en los diccionarios anteriormente descritos. El Diccionario Genérico contiene un total de 6 elementos clave objetivo, el Diccionario Automóvil contiene 8 elementos y el Diccionario Hogar contiene 9. En la póliza de seguro

del automóvil, del total de 14 elementos clave objetivo, la herramienta es capaz de extraer correctamente 12 de ellos. En la póliza de seguro del hogar de los 15 elementos clave objetivo, la herramienta solo extrajo correctamente 2 de ellos. Sin embargo, en el caso de esta póliza, muchos de los elementos clave no se encontraban en el documento. En concreto 9 de los 15 elementos clave objetivo estaban presentes. Una vez más, la razón por la que los resultados son tan pobres es el mal rendimiento de la extracción de texto, ya que al no tener el texto la herramienta no podrá encontrar la similitud entre los elementos del texto y los diccionarios.

DICCIONARIO GENÉRICO - SEGURO DEL AUTOMÓVIL	
Policy number	12345-67-89
Name insured	Jane Smith
Effective date	01/01/2019 0:01
Premium	3797.6
Address	1234 Main St
Limit	$250K each person/$800K each incident

Tabla 5.2. ElementosClave.csv (1) generado por la herramienta.

DICCIONARIO ESPECÍFICO - SEGURO DEL AUTOMÓVIL	
Model	N/A
Principal driver	N/A
Comprehensive deductible	3500
Collision deductible	$1,000
Bodily injury	$250K each person/$800K each incident
Property damage	$100K each incident
Medical coverage	$5,000 each person
Uninsured motorist bodily injury	$250K each person/$500K each incident

Tabla 5.3. ElementosClave.csv (2) generado por la herramienta.

5.3 COMPARATIVA DE LOS RESULTADOS

Una vez analizados los resultados que ofrecen ambas soluciones, se va a proceder a la comparativa de los mismos.

En primer lugar, en la tarea de extracción del texto, la herramienta de Google sobresale sin duda, obteniendo una tasa de acierto cercana al 100% en ambas pólizas, frente al 89% y 60%

de tasa de acierto de la herramienta desarrollada en este TFG. Se puede observar aquí una flaqueza de dicha herramienta que debe ser mejorada. En segundo lugar, en la tarea de extracción de relaciones entre elementos, Document AI de Google ofrece resultados muy pobres, estableciendo un número pequeño de relaciones que en muchas ocasiones no llegan a tener sentido. En la póliza de seguro del automóvil, la herramienta de este TFG formó 25 relaciones válidas, frente a las 18 de Google. En la póliza de seguro del hogar, la herramienta de este TFG encontró 10 relaciones, frente a las 2 relaciones válidas encontradas por Document AI. Por último, en la tarea de extracción de elementos clave, no existe comparativa posible, ya que la herramienta de Google no ofrece este servicio.

Se puede concluir que la herramienta de Google incorpora una gran herramienta de OCR, pero que tiene problemas a la hora de formar relaciones y no es capaz de extraer elementos clave. En cambio, la herramienta de extracción automática de datos desarrollada en este TFG establece de manera satisfactoria relaciones entre elementos del documento y es capaz de extraer gran cantidad de los elementos clave objetivo; sin embargo, tiene problemas a la hora de extraer el texto con precisión.

Capítulo 6. VIABILIDAD ECONÓMICA

En este breve capítulo se va a presentar un modelo de negocio basado en ofrecer la herramienta desarrollada en este Trabajo de Fin de Grado como servicio en la nube para la extracción de datos de documentos. Se introducirá una idea de Producto Mínimo Viable (*MVP*) y se presentarán los cálculos de ingresos y costes de este posible negocio.

De igual manera que Amazon, Google y Microsoft ofrecen sus servicios de extracción de datos de documentos en forma de servicio de pago, el modelo de negocio de este TFG está basado en este mismo servicio. Los clientes podrán extraer todo el texto encontrado en sus documentos, extraer relaciones entre los elementos encontrados y extraer elementos clave de ellos. Para que el servicio esté siempre disponible para los potenciales clientes será necesario que la herramienta esté alojada en la nube. Para ello existen multitud de servicios disponibles; una vez más, estos servicios son ofrecidos principalmente por Amazon, Google y Microsoft. La estimación de costes e ingresos se va a realizar suponiendo que el servicio de procesamiento en la nube es contratado a AWS y que el precio que se cobra al cliente por el uso de la herramienta desarrollada en este TFG como máximo el cobrado por Amazon Textract en este mismo servicio. Los cálculos han sido realizados suponiendo que 1.000.000 de páginas de documentos serán analizadas con la herramienta.

Coste de procesamiento en la nube con AWS

Para ofrecer el servicio descrito sería necesario contratar tres productos de AWS. En primer lugar, sería necesario un Amazon EC2, donde se almacenaría la herramienta y se procesarían los documentos. En segundo lugar, serían necesarios dos *Amazon Simple Storage Service*, o S3, uno donde el cliente pueda subir los documentos que desea almacenar y otro donde el cliente pueda retirar los archivos csv generados. A continuación, se detallan los costes de contratar dichos productos. La informa de dichos costes ha sido calculada con la herramienta 'Pricing Calculator' de AWS (https://calculator.aws/#/).

La instancia EC2 contaría Linux con sistema operativo. En esta instancia se instalaría Python y clonaría el repositorio de GitHub con la herramienta. Dado que el modelo ya estaría entrenado y afinado, no es necesario contar con un gran poder de procesamiento. Por ello, se ha elegido la versión más sencilla de instancia EC2, llamada 't4g.xlarge'. Esta instancia cuenta con un procesador CPU de 4 núcleos y 16 GB de almacenamiento RAM. Contratar este servicio supone un coste mensual de 69,28 USD.

Para poder almacenar la herramienta es necesario que a la instancia EC2 se le asigne cierto espacio de almacenamiento. Es por ello, que es necesario un 'Amazon Elastic Block Storage (EBS)'. Para almacenar la herramienta 30 GB son suficientes. Esto supone un coste adicional de 3,48 USD.

Por un lado, el tamaño medio de las imágenes de pólizas de seguros usadas para el análisis de la herramienta es de 120 kB. Esto supone que se harían mensualmente 1.000.000 de escrituras de 120kB de media, o 120 GB mensualmente. Por otro lado, el tamaño conjunto de los tres archivos *csv* nunca supera los 5 kB. Por lo que, 1 millón de escrituras mensuales supondrían 5 GB/mes. Para satisfacer esta demanda, el S3 contratado dedicado a que los clientes suban sus documentos tendrá un coste de 6,33 USD mensuales. El S3 dedicado a que los clientes se descarguen los csv generados por la herramienta tendrá un coste de 5,78 USD/mes.

Los costes totales por contratar los servicios de procesamiento y almacenamiento en la nube de AWS son de 78,54 USD.

Precios de extracción de texto, extracción de relaciones y extracción de elementos clave con Amazon Textract

Se va a tomar la herramienta de Amazon *Textract* como referencia de precio al que se podría cobrar el servicio ofrecido.

Con la herramienta de Amazon la extracción de texto tiene un coste de 1,5 USD por cada 1.000 páginas.

*Detectar texto: 1.000 * 1,5 USD = 1.500 USD*

La extracción de relaciones de la herramienta desarrollada en este TFG equivale al servicio de *Textract* de 'Extracción de formularios'. Así mismo, la extracción de elementos clave equivale a su servicio de 'Extracción basada en consultas'. En la Figura 6.1 se puede observar que el precio conjunto de extraer formularios y consultas es de 55 USD cada 1.000 páginas,

*Analizar formularios y consultas = 1.000 * 55 USD = 55.000 USD*

La suma de los tres servicios tendría un coste de 66.500 USD para el cliente.

		Primer millón de páginas en un mes
API para detectar texto de un documento	Cada 1000 páginas	1,50 USD

API para analizar documentos		**Primer millón de páginas en un mes**
Consultas	Cada 1000 páginas	15,00 USD
Tablas	Cada 1000 páginas	15,00 USD
Tablas y consultas	Cada 1000 páginas	20,00 USD
Formularios	Cada 1000 páginas	50,00 USD
Formularios y consultas	Cada 1000 páginas	55,00 USD
Tablas y formularios	Cada 1000 páginas	15,00 USD + 50,00 USD
Tablas, formularios y consultas	Cada 1000 páginas	70,00 USD

Figura 6.1. Tabla de costes del servicio de Amazon Textract. Extraída de https://aws.amazon.com/es/textract/pricing/.

Conclusión

Cómo se puede ver por lo números, la extracción de datos de documentos cómo servicio es un negocio muy lucrativo. Sin embargo, no se puede equiparar la herramienta desarrollada en este TFG con los servicios ofrecidos por Amazon, ya que la empresa no ofrece Textract como servicio aislado, sino que puede ser integrado con muchos otros servicios que juntos aportarán verdadero valor a las empresas. Por ello, no sería realista que la herramienta de este TFG tuviese el mismo precio que Textract. Sin embargo, los márgenes son los suficientemente grandes para que, aun cobrando una décima parte a los clientes, el negocio siga siendo lucrativo.

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

7.1 CONCLUSIONES

Una vez finalizado el desarrollo de la herramienta propuesta en este Trabajo de Fin de Grado se procede a listar una recopilación de los logros conseguidos.

En primer lugar, se ha podido observar la creciente demanda de herramientas de extracción de datos por parte de las empresas. Observando crecimientos anuales en el número de usuarios de hasta el 100%, como es el caso de la plataforma de Google, Document AI.

En segundo lugar, se ha probado la eficacia de los modelos basados en *Transformers* y método de entrenamiento basado en pre-entrenamiento más afinado. Gracias a la arquitectura de los *Transformers* y su mecanismo de autoatención, estos modelos son capaces de entender el contexto en el que se encuentra cada palabra y adquirir un mejor conocimiento del lenguaje. También se ha podido comprobar la versatilidad de esta arquitectura, permitiendo crear una gran variedad de modelos con distintos propósitos.

En tercer lugar, se ha comprobado cómo introducir elemento visuales y de posición aumenta el rendimiento de los modelos en las tareas de *Document AI*. Elementos visuales que informen al modelo sobre el diseño y formato del documento son primordiales para que entienda bien un documento. Esto puede observarse gracias a que la mayoría de los modelos que encabezan los rankings de rendimiento en las principales tareas del *Document AI* están basados en modificaciones de la arquitectura *Transformers* que incorporan elementos visuales.

En cuarto lugar, se ha observado que una herramienta precisa de OCR es un elemento clave y al que se le debe dar prioridad ya que una extracción poco precisa del texto supondrá un peor rendimiento en del resto de tareas en general.

Finalmente, se ha comprobado lo rentable y lucrativo que es un negocio basado en la extracción de datos de documentos, gracias a unos costes de operación muy bajos.

7.2 TRABAJOS FUTUROS

Con el objetivo de en un futuro mejorar los resultados ofrecidos por la herramienta desarrollada en este Trabajo de Fin de Grado se van a listar a continuación una serie de posibles trabajos futuros.

En primer lugar, como ya se ha comentado previamente, en abril de 2022 salió a la luz el modelo LayoutLMv3 (Huang, Lv, Cui, Lu, & Wei, 2022). Este modelo estableció un nuevo estado del arte en los principales *benchmarks*, superando el rendimiento de la versión utilizada en la herramienta desarrollada, LayoutLMv2. Por ello, resultaría interesante la implementación de este nuevo modelo en la herramienta. Esto podría mejorar las predicciones en el etiquetado {B,I,E,S,O} e incluso, según la información encontrada en el artículo, evitar la necesidad de funciones que formen frases y párrafos, ya que propio modelo agruparía los *tokens* para formarlas.

En segundo lugar, la adaptación de la herramienta a otros idiomas, principalmente español, podría resultar interesante. La principal razón por la que la herramienta de este TFG analiza solamente documentos en inglés es porque el modelo LayoutLMv2 solo admite dicho idioma. Sin embargo, existe una versión de esta familia de modelos que si permite otros idiomas, LayoutXLM (Xu, y otros, 2021). Para la implementación de este modelo se deberán utilizar también *datasets* similares FUNSD, pero en el idioma que se desee.

En tercer lugar, tal y como se ha visto en el 'Capítulo 5: Análisis de Resultados', la herramienta desarrollada en este TFG tiene dificultades para extraer de manera precisa el texto encontrado en los documentos. La mejora de los resultados obtenidos en esta tarea en un futuro es importante, ya que una mejora en esta tarea supondrá también una mejora en las tareas de extracción de relaciones y extracción de elementos clave.

Por último, puede ser interesante la implementación de un algoritmo que permita analizar documentos con diversas páginas y/o que permita dividir automáticamente un documento cuando contenga más elementos de los permitidos por el modelo.

Una vez implementadas todas estas mejoras y cuando se considere que la herramienta ofrece resultados satisfactorios, sería conveniente implementar una función que permita analizar lotes de documentos.

Capítulo 8. BIBLIOGRAFÍA

Amazon Web Services. (s.f.). ¿Qué es el reconocimiento óptico de caracteres (OCR)? Obtenido de https://aws.amazon.com/es/what-is/ocr/. Último accesso: 23/08/2022.

Cui, L., Xu, Y., Lv, T., & Wei, F. (2021). *Document AI: Benchmarks, Models and Applications.* arXiv:2111.08609v1.

Data Semantics. (s.f.). *Advantages of Using an Intelligent Document Processing Tool.* Obtenido de https://datasemantics.co/advantages-of-intelligent-document-processing/. Último acceso: 25/08/2022.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transfromers for Languaje Understanding.* arXiv:1810.04805v2.

Grafiada, J. A. (2022). ¿Qué es un seguro? Obtenido de https://www.rankia.com/blog/mejores-seguros/2449635-que-seguro-tipos-seguros-existen. Ultimo acceso: 09/08/2022.

Harley, A. W., Ufkes, A., & Derpanis, K. G. (2015). *Evaluation of Deep Convolutional Nets for Document Image Classification and Retrieval.* International Conference on Document Analysis and Recognition (ICDAR). Celebrada en Nancy, Francia, 23-26 agosoto 2015. Oral session 5. ICDAR 2015 Booklet, página 27.

Hmrishav, B. (2022). *What Is Computer Vision? [Basic Tasks & Techniques]*. v7labs. Obtenido de https://www.v7labs.com/blog/what-is-computer-vision#h3. Último acceso: 25/08/2022.

Huang, Y., Lv, T., Cui, L., Lu, Y., & Wei, F. (2022). *LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking.* arXiv:2204.08387.

Huang, Z., Chen, K., He, J., Bai, X., Karatzas, D., Lu, S., & Jawahar, C. (2021). *ICDAR 2019 Competition on Scanned Receipt OCR and Information Extraction.* arXiv:2103.10213v1.

IBM. (2021). ¿Qué es la Visión Artificial? Obtenido de https://www.ibm.com/es-es/topics/computer-vision. Último acceso: 20/08/2022.

IBM Cloud Education. (2020a). Deep Learning. Obtenido de https://www.ibm.com/cloud/learn/deep-learning. Último acceso: 24/09/2022.

IBM Cloud Education. (2020b). Natural Languaje Processing (NLP). Obtenido de https://www.ibm.com/cloud/learn/natural-language-processing. Último acceso: 14/09/2022.

ICDAR. (s.f.). *ICDAR*. Obtenido de https://www.icdar.org/. Último acceso: 22/08/2022.

IEEE. (2020). Top Programming Languages 2020. Obtenido de https://spectrum.ieee.org/top-programming-language-2020. Último acceso: 10/08/2022.

Jaume, G., Ekenel, H. K., & Thiran, J.-P. (2019). *FUNSD: A Dataset for Form Understandingin Noisy Scanned Documents.* arXiv:1905.13538v2.

Kanapala, A., Pal, S., & Pamula, R. (2019). *Text summarization for legal documents: a survey.* Springer.

LeCun, Y. (2021). Self-supervised learning: The dark matter of intelligence. (M. AI, Ed.) Obtenido de https://ai.facebook.com/blog/self-supervised-learning-the-dark-matter-of-intelligence/. Último acceso: 24/09/2022.

Li, M., Cui, L., Huang, S., Wei, F., Zhou, M., & Li, Z. (2020). *TableBank: A Benchmark Dataset for Table Detection and Recognition.* arXiv:1903.01949v2.

Mathew, M., Karatzas, D., & Jawahar, C. (2021). *DocVQA: A Dataset for VQA on Document Images.* arXiv:2007.00398v3.

Naciones Unidas. (2015). *Objetivos de Desarrollo Sostenible.* Obtenido de https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/. Último acceso: 29/08/2022.

Park, S., Shin, S., Lee, B., Lee, J., Surh, J., Seo, M., & Lee, H. (2019). *CORD: A Consolidated Receipt Dataset for Post-OCR Parsing.* Obtenido de https://openreview.net/forum?id=SJl3z659UH

Radečić, D. (2020). Google Colab: How does it compare to a GPU-enabled laptop? (T. D. Science, Ed.). Último acceso: 12/08/2022.

Reimers, N., & Gurevych, I. (2019). *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.* arXiv:1908.10084v1.

Ren, S., He, K., Girshick, R., & Sun, J. (2016). *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.* arXiv:1506.01497v3.

Sakira, N., Sirisala, N., & Velpuru, M. S. (2021). *CNN based Optical Character Recognition and Applications.* IEEE. Obtenido de https://ieeexplore.ieee.org/abstract/document/9358735/authors#authors

Unespa. (2018). *EL SEGURO EN LOS HOGARES. ENCUESTA DE PRESUPUESTOS 2018. INE.* Obtenido de https://www.unespa.es/main-files/uploads/2020/06/1.1.c.-El-seguro-en-los-hogares-EPF-IES-2019-FINAL.pdf

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). *Attention Is All You Need.* arXiv:1706.03762.

Walker, A. (2021). Document Searches Can Waste 100s of Hours. Obtenido de https://insurance-edge.net/2021/11/10/document-searches-can-waste-100s-of-hours/. Último acceso: 25/08/2022.

Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., & Zhou, M. (2020). *MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers.* arXiv:2002.10957v2.

Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., & Zhou, M. (2020). *LayoutLM: Pre-training of Text and Layout for Document Image Understanding.* arXiv:1912.13318v5.

Xu, Y., Lv, T., Cui, L., Wang, G., Lu, Y., Florencio, D., . . . Wei, F. (2021). *LayoutXLM: Multimodal Pre-training for Multilingual Visually-rich Document Understanding.* arXiv:2104.08836v3.

Xu, Y., Xu, Y., Lv, T., Cui, L., Wei, F., Wang, G., . . . Zhou, L. (2021). *LayoutLMv2: Multi-modal Pre-training for Rich Document Understanding.* arXiv:2012.14740v3.

Zhang, J., Zhao, Y., Saleh, M., & Liu, P. J. (2020). *PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization.* arXiv:1912.08777v3.

Zhong, X., Tang, J., & Jimeno, A. (2019). *PubLayNet: largest dataset ever for document layout analysis.* arXiv:1908.07836v1.

ANEXO I: ALINEACIÓN CON LOS ODS

Vivimos en un mundo globalizado en el que todo el mundo sabe que está ocurriendo en cada rincón del mundo. No sería humano mirar hacia otro sabiendo los problemas existentes, especialmente en los países menos desarrollados. Es por ello que, en septiembre de 2015, los líderes mundiales adoptaron un conjunto de objetivos globales para erradicar la pobreza, proteger el planeta y asegurar la prosperidad (Naciones Unidas, 2015). Cada objetivo tiene metas específicas que deben alcanzarse en los próximos 15 años.

Estos Objetivos de Desarrollo Sostenible sirven a modo de marco moral para que cada proyecto o iniciativa llevada a cabo se desarrolle con ellos en mente, especialmente en un área con potencial suficiente de generar un impacto positivo, como es la ingeniería.

Este Trabajo de Fin de Grado se alinea principalmente con los Objetivos 8 y 9.

El Objetivo 8 se centra en 'promover el crecimiento económico sostenido, inclusivo y sostenible, el empleo pleno y productivo y el trabajo decente para todos'. Este TFG ayuda especialmente a la consecución de las metas 8.2 y 8.3, las cuales pretenden lograr mayor productividad económica mediante la innovación tecnológica e innovación y, fomentar el crecimiento de pequeñas y medianas empresas, respectivamente. La herramienta desarrollada permite el análisis de documentos y la extracción de datos a pequeñas y medianas empresas que no puedan permitirse la inversión en desarrollo informático. Al no requerir de conocimientos de programación para ser usada, ni de grandes cantidades de documentos anotados manualmente para ser entrenada, cualquier usuario con cualquier nivel de conocimiento informático será capaz de utilizarla. De este modo, se estará dando acceso a todo el mundo a una herramienta que ayuda a lograr mayor productividad empresarial, fomentando así el crecimiento.

El Objetivo 9 tiene el propósito de 'construir infraestructuras resilientes, promover la industrialización inclusiva y sostenible y fomentar la innovación'. En concreto, la herramienta desarrollada se alinea con las metas 9.5 y 9.b. La meta 9.5 tiene el objetivo de mejorar la capacidad tecnológica de los sectores industriales, mientras que la meta 9.b se centra apoyar al desarrollo de tecnologías, la investigación y la innovación en los países de desarrollo. La herramienta desarrollada consigue ayudar al cumplimiento de estas metas al poner en manos de todo tipo de usuarios una herramienta de extracción automática de datos hecha con modelos de Inteligencia Artificial.

ANEXO II

En este breve anexo de muestran figuras, ordenadas de izquierda a derecha, del proceso de análisis de las dos pólizas de seguros analizadas en el 'Capítulo 5: Análisis de los resultados'.

Figuras II.1-6. Proceso de análisis de una póliza de seguro del automóvil.

Figuras II.7-12. Proceso de análisis de una póliza de seguro del hogar.

ANEXO III

En el siguiente enlace se puede encontrar el repositorio de GitHub donde está alojada la herramienta desarrollada en este TFG. En este repositorio también se pueden encontrar ejemplos de resultados del procesado de alguna póliza de seguro.

https://github.com/paavlito/TFG_PabloGarciaBolivar