Extracción de Información de Facturas(ar) Agrupada en Jerarquías de Negocios

Marcos Maciel[1], Claudia Pons[2]
[1] CAETI UAI, Buenos Aires, Argentina
Mmaciel03@hotmail.com
[2] Universidad Nacional de La Plata, UAI, Buenos Aires, Argentina
Claudia.pons@uai.edu.ar

Resumen. Las compañías intercambian una gran cantidad y variedad de facturas (ar) en formato digital e incluso en formato de papel. La extracción automática de información se vuelve compleja por la diversidad en las extensiones y diseños de estos documentos. El objetivo de extraer información es alimentar procesos de negocios con datos clasificados y que cumplan con parámetros de calidad. Para abordar la complejidad se propone un pipeline que integra un modelo de inteligencia artificial y un modelo de programación tradicional capaz de extraer información en jerarquía de negocios y en un segundo paso procesarla en un motor de reglas para cumplir con parámetros de calidad personalizados. En este artículo se presenta la extracción de información de facturas argentinas mediante la clasificación de entidades con el modelo de inteligencia artificial preentrenado LayoutLM, conversión del resultado en una jerarquía de negocios para reutilizarlo en un motor de reglas con capacidad de optimizar la calidad de la información y robustecer los procesos de negocios, así demostramos que el uso integrado de programación con inteligencia artificial y programación tradicional es una solución superadora al uso en forma independiente.

Keywords: Machine Learning, Layout, Document image understanding, Invoice, Rule Engine.

1 Introducción

Las compañías trabajan a diario intercambiando facturas por servicios y/o productos, la información contenida de estos documentos se presentan en múltiples formatos como ser archivos digitalizados tipo PDF[1] o imágenes jpg, png, jpeg etc., pero también en papel. El origen de estos documentos es entre otros: archivos originales generados por entidades públicas o empresas terceras privadas, archivos emitidos in-house en empresas privadas, copias de archivos escaneados, fotocopias o incluso fotos tomadas con dispositivos móviles, etc. La diversidad de diseños y extensiones en los documentos hace que la extracción de información automática de forma genérica sea una tarea compleja [17], el procesamiento manual es lento, propenso a errores y el costo se incrementa cuando la cantidad de facturas crece [15].

A pesar de lo anterior, disponer de la información de estos documentos en forma

[1]https://www.adobe.com/la/acrobat/about-adobe-pdf.html

rápida y automática es de suma importancia para los usuarios del negocio [27], ellos usan el contenido de estos documentos como insumo de otros procesos, para enriquecer nuevos modelos de negocios y/o para la toma de decisiones [13].

Para dar respuestas a lo antes mencionado se usan modelos preentrenados de inteligencia artificial que clasifican y predicen información dado un conjunto de datos previamente etiquetado. Estos modelos almacenan un conocimiento probado de manera empírica sobre una tarea específica [9], fueron entrenados con gran volumen de datos haciendo uso de amplios recursos computacionales [25], y permiten transferir el conocimiento previamente adquirido de un dominio a otro [18, 26, 9, 6]. Entre las ventajas de usar esta alternativa se encuentran contar con un marco de referencia, ahorrar tiempo y recursos, transferir el conocimiento almacenado (Transfer Learning) e implementar cambios sobre un modelo entrenado (Fine-Tuning) [9] y evita comenzar desde cero.

En este trabajo se propone la extracción de información clasificada de facturas argentinas de forma automática usando el modelo preentrenado LayoutLM de Microsoft [29] y la agrupación de la predicción en jerarquía de entidades de negocios para usarla como insumo en un motor de reglas con el objetivo de enriquecer, robustecer y potenciar el conocimiento. Para lograr estos objetivos se hace un Transfer Learning compuesto de 2 fases: capturar el conocimiento adquirido por LayoutLM y en segundo lugar hacer Fine-Tuning (ajustes) para transferir el aprendizaje a nuestra clasificación.

Para el alcance de este artículo se restringe el conjunto de datos a 5 diseños heterogéneos de facturas argentinas y 7 entidades de negocios, el aporte de este trabajo consiste en un reentrenamiento del modelo LayoutLM con entidades y clases personalizadas pertenecientes al dominio de facturas argentinas, para esto se agregan dos capas extras: (a) una para filtrar las entidades seleccionadas y asignación de etiquetas incluyendo además un preprocesamiento para optimizar la calidad de los datos usados para entrenamiento y validación, (b) por último se agrega una capa responsable de jerarquizar la predicción del modelo agrupando las entidades por semántica del dominio. En Fig. 1 se presentan ambas capas (a) Custom Entity Preprocessing y (b) Hierarchical Organization.

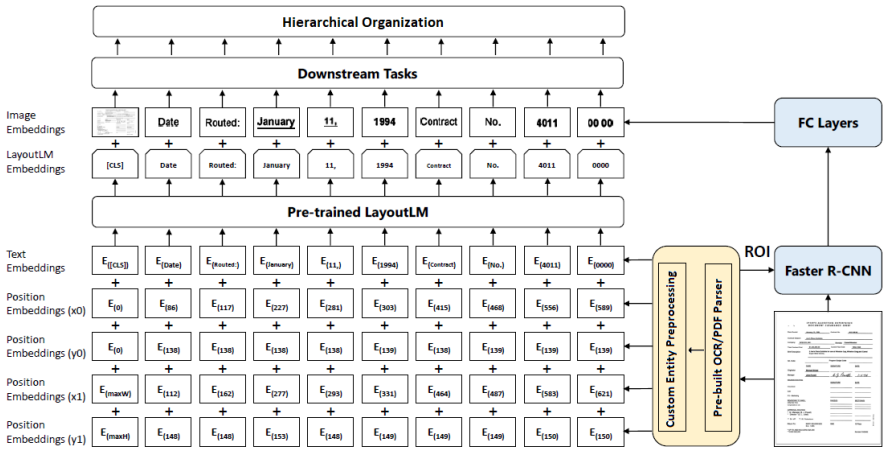

Fig. 1. Arquitectura LayoutLM Microsoft con dos capas adicionales.

Este artículo está organizado de la siguiente forma: trabajos relacionados en la sección 2, en la sección 3 explicamos la fase de preprocesamiento incluyendo un detalle del conjunto de datos propuesto, el etiquetado de entidades con interés para este trabajo y la fase de entrenamiento con LayoutLM. Los resultados son presentandos en la sección 4, en la sección 5 se combinan las entidades semánticas extraídas de las facturas con programación tradicional para robustecer los procesos de negocios. Por último se presentan conclusiones y trabajo futuro en la sección 6.

2 Trabajos relacionados

La extracción de información de documentos entre ellos facturas y recibos es materia de investigación actualmente debido a la dificultad de clasificar una entidad en un contexto de negocios, a continuación, se enumeran trabajos presentados en esta línea de investigación.

Bardelli exploró la extracción del VAT (Value-added Tax) en compañías italianas desde archivos XML, propuso un pipeline para construir un conjunto de datos y aplicar modelos de IA para predecir 2 variables objetivos [2]. Los autores concluyeron que una variable objetivo presenta más problemas que otras debido a su alto número de etiquetas posibles, una clasificación jerárquica que refleje el catálogo de cuentas es considerado una posible solución. La extracción de información de facturas húngaras fue presentada en [23], Decision Tree Classifier, Random Forest Classifier y Extreme Gradient Boosting Classifier fueron implementados sobre atributos etiquetados por su contexto dentro del documento. El aporte de este trabajo se basó en agregar información contextual a las clases objetivos.

Usando un modelo existente basado en un transformador pre-entrenado redujo la necesidad de documentos anotados, esta aproximación permite recuperar cualquier esquema de información estructurada mediante un modelo adaptado sequence-to-sequence (convertir secuencias de un dominio a otro) basado en atención [19]. Kleister usa características textuales y diseño estructural de los documentos para extraer información mediante NLP (Natural Language Processing) sobre documentos extensos [7, 21], estos dos últimos trabajos fueron aplicados a documentos no español.

A una lista de características extraídas con un OCR (Optical Character Recognition) etiquetada con 8 clases, tres modelos Naive Bayes, Logistic Regression y SVM (Support Vector Machine) fueron empleados para clasificar cada entidad, el trabajo logró 13.99% de error en testing y concluyó que SVM fue el modelo que mejor predicción obtuvo [16], en estos casos los modelos no tiene capacidad de aprender ya que apuntan a clasificar datos.

Una arquitectura modular para evaluar la detección de tipo de bloques de textos basados en análisis de texto combinado con características de diseño de facturas es presentado en [30] alcanzando una tasa de predicción del 80.1% inferior a la tasa de predicción de LayoutLM.

3 Extracción de información de factura

La primera etapa consiste en construir un conjunto de datos con facturas argentinas, en este caso al ser información del ámbito local a este trabajo y con datos sensibles los ejemplos adjuntos están ocultos o ligeramente modificados. Las facturas y/o recibos aportados para la investigación fueron 6.000 archivos digitalizados, una clasificación inicial permitió determinar el origen según: documentos con extensión .pdf generados directamente en la entidad recaudatoria Argentina [1] con diseños heterogéneos subordinado por la condición fiscal del emisor [5], documentos Word[2] con imágenes insertadas, imágenes creadas con cámaras de teléfonos móviles y resolución variada, documentos generados por sistemas propietarios con diseños personalizados. Una particularidad que afecta a todos los casos es que los documentos argentinos repiten entidades claves como por ejemplo CUIT (Clave única de identificación tributaria), domicilio, nombre y apellido o razón social, condición frente a IVA (Impuesto al valor agregado) para emisores y receptores haciendo más compleja la extracción y jerarquización de atributos por clases.

El siguiente paso abarca la fase de extracción de datos de cada documento con un OCR y el modelado para adaptarlo a las necesidades del modelo de inteligencia artificial seleccionado. El modelado incluye una etapa de etiquetado, esta tarea de anotación manual del contenido de documentos es catalogada de alto costo debido es un proceso que consume mucho tiempo y es propenso a errores [4, 24]. En consecuencia, el proceso de etiquetado para este trabajo se dividió en un primer paso de anotación automática y un segundo paso de revisión y ajustes manual. Para este artículo etiquetar implica asignar una clase -con una semántica orientada a negocios - de forma correcta a una entidad o al valor que se corresponde a una entidad usando el contenido de un documento. Por último, usamos LayoutLM porque es un modelo que combinando texto y la estructura del documento logró una tasa de acierto superior al 90% [29], con pocos documentos anotados tiene una tasa superior al 80%, es apropiado para trabajar sobre nuevos conjuntos de datos con pequeños cambios [20] y porque la cantidad límite de 512 palabras del transformer preconfigurado [29, 22, 3] es suficiente para nuestro interés.

3.1 Preprocesamiento y producción del conjunto de datos

El conjunto de datos para entrenar y validar el modelo fue construido a partir de un lote igual a d=5.148 archivos, entre los que se encontraron extensiones como por ejemplo pdf, jpg, png, gif etc. Aunque existen una extensa variedad de diseños de facturas, para este trabajo fue limitada la cantidad a t=5 diseños no uniformes. Dos diseños de facturas son presentados en la figura 2, datos sensibles fueron excluidos por política de privacidad. Se pueden mencionar otras categorías de impuestos, por ejemplo: facturas A-B-C, recibos, factura de crédito electrónica con detalle de tributos Afip y nota de crédito.

El primer paso fue dividir el lote de archivos entre pdf e imágenes y a las imágenes por tipo de extensión. A la lista de archivos pdf se los procesa con PyPDF2[3] para recortar solo la primera página ya que muchos ejemplos fueron encontrados con 3 hojas para original, duplicado y triplicado con la misma información excepto esta palabras como título, para minimizar el tiempo de procesamiento de imágenes se trabaja solo con la primera hoja. Por último, todos los archivos fueron convertidos a imágenes de extensión jpg aplicando escala de grises que ayude a disminuir la complejidad del entrenamiento con la librería fitz[4], el resultado final es d=5.148 archivos de imágenes.

[2]https://www.microsoft.com/es-es/microsoft-365/Word
[3]https://pypdf2.readthedocs.io/en/latest/
[4]https://pymupdf.readthedocs.io/en/latest/module.html

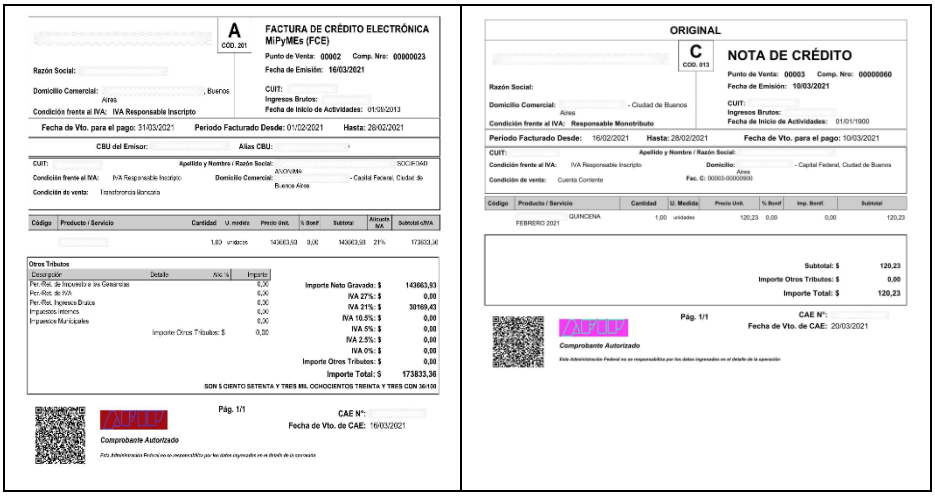

Fig. 2. Diseño de facturas argentinas.

La segunda etapa consistió en extraer de cada imagen su información basada en palabras y su ubicación. Nos referimos a ubicación como un rectángulo que encierra una palabra y tiene una dirección única en el documento determinada por su distancia con respecto al margen superior, al margen izquierdo y el ancho-alto de la palabra como se puede ver en la figura 3. Para este proceso se usó un OCR cuyo nombre es pytesseract[5]. Esta librería retorna una estructura tipo tabla (dataframe en Pandas[6]) de columnas = 12 y registros = cantidad de lecturas reconocidas. Otras columnas fueron agregadas para ejecutar el proceso de transformer [10] a saber: {name, img_width, img_heigth, label} o {nombre_archivo, ancho_imagen, alto_imagen, clase}.

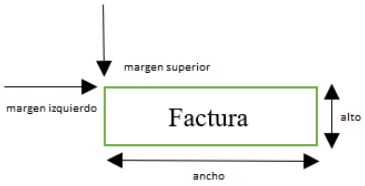

Fig. 3. Ejemplo de un rectángulo o bounding box.

Por cada imagen se ejecuta la rutina en Python[7] desarrollada para esta tarea y el resultado es almacenado como nuevo archivo con extensión .csv separado por punto y coma.

[5]https://github.com/tesseract-ocr/tesseract
[6]https://pandas.pydata.org/
[7]https://www.python.org/

3.2 Etiquetado de entidades semánticas

Partiendo del punto anterior con el lote de datos depurados las palabras de interés fueron etiquetadas mediante un proceso híbrido que incluye un etiquetado automático y una validación visual junto a corrección manual. Un corpus compuesto con las entidades de negocios a extraer y su clase asignada fue creado, este corpus incluye una entidad O(otros) para el resto de las palabras sin interés. La tabla 1 expone la relación entre entidad-clase y entidad-valor-clase.

Tabla 1. Listado de entidades de negocios y clases asignadas.

Entidad	Clase
Tipo de factura	INVOICE_TYPE_ID
Punto de ventas	SELL_POINT_ID
Punto de venta valor	SELL_POINT_NUMBER
Fecha emisión	ISSUE_DATE_ID
Fecha emisión valor	ISSUE_DATE
Cuit	CUIT_SELLER_ID/ CUIT_CLIENT_ID
Cuit valor	CUIT_SELLER/ CUIT_CLIENT
Importe facturado	TOTAL_AMOUNT_ID
Importe facturado valor	TOTAL_AMOUNT
Número cae	CAE_ID
Número cae valor	CAE_NUMBER
Otros valores	O

El etiquetado automático responde a un programa personalizado para este trabajo desarrollado con Python, que fue posible porque la extracción de información desde los archivos de imágenes con la librería de OCR retorna las palabras y sus valores asignados respetando un cierto orden de inserción en el documento que va de la esquina superior izquierda en dirección positiva al eje x y descendente en altura.

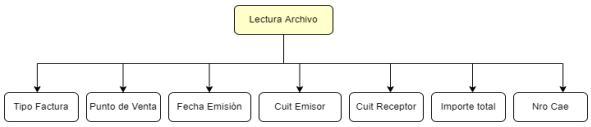

Fig. 4. Lista de entidades de negocios.

La rutina de auto etiquetado en Python crea una colección de datos de cada archivo (.csv) procesado en el punto 3.1, cada entidad es recuperada de acuerdo con el orden de la figura 4. Luego, por cada coincidencia C=entidad [palabras (tabla 1)], una nueva colección es creada para contener la entidad y su posible valor N[x]= [palabra, clase] ver figura 5. El etiquetado responde a un esquema de anotación llamado BIOES (ej. figura 5) por ser un esquema más expresivo y mejora la performance del modelo comparado al esquema IOB [31], donde cada palabra puede ser inicio B(beginning), interior I(inside), apertura S(start), fin E(end) y aquellas sin interés de reconocimiento o fuera de alcance O(outside). El proceso de anotación finaliza con una validación visual y corrección manual de errores encontrados.

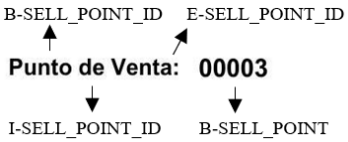

Fig. 5. Anotación de entidad y clase.

3.3 Modelo Layout Microsoft

El modelo seleccionado es LayoutLM de Microsoft porque unifica la interacción entre texto e información de diseño de imágenes además de su tasa de acierto superior al 90% [29]. Para construir las matrices de elementos multidimensionales o tensores[8] es necesario contar con 3 archivos para entrenamiento y 3 archivos de validación, estos archivos fueron creados con una rutina Python que toma cada archivo con extensión csv resultado del preprocesamiento y etiquetado del paso anterior. Esta rutina lee las columnas {left= distancia magen izquiero, top= distancia margen superior, width = ancho texto, height = alto del texto, text= texto reconocido, name= nombre del archive procesado, img_width = ancho imagen procesada, img_heigth = alto imagen procesado, label= etiqueta asignada} y crea {train-test}.txt con las columnas text y label (ejemplo tabla 2), crea {train-test}_box.txt con las columnas text y left, top, width, height normalizado con valores entre 0-1000 (ejemplo tabla 3), crea {train-test}_image.txt con las columnas text, left, top, width, height, img_width, img_heigth, name (ejemplo tabla 4).

Tabla 2. Formato del archivo train.txt y test.txt

Campo	Etiqueta
factura	S-INVOICE_TYPE_ID
de	B-INVOICE_TYPE_ID
crédito	I-INVOICE_TYPE_ID
electrónica	I-INVOICE_TYPE_ID
mipyme:	I-INVOICE_TYPE_ID
apellido vendedor	O
nombre vendedor	O
(fce)	E-INVOICE_TYPE_ID

Tabla 3. Formato del archivo train_box.txt y test_box.txt

Campo	x izq	y superior	ancho	alto
factura	434	49	511	58
de	518	49	538	58
crédito	546	49	616	58
electrónica	427	66	542	75
mipyme:	548	66	625	78
apellido vendedor	63	75	141	90
nombre vendedor	152	75	249	90
(fce)	508	82	547	95

Tabla 4. Formato del archivo train_image.txt y test_image.txt

Campo	x izq	y sup	x der	y inf	ancho	alto	nombre imagen
factura	2070	334	2437	394	4760	6728	0000000194.png

[8]https://pytorch.org/docs/stable/tensors.html

de	2467	334	2563	394	4760	6728	0000000194.png
crédito	2599	334	2935	394	4760	6728	0000000194.png
electrónica	2034	448	2581	508	4760	6728	0000000194.png
mipyme:	2611	448	2977	526	4760	6728	0000000194.png
apellido vendedor	300	508	672	610	4760	6728	0000000194.png
nombre vendedor	726	508	1188	610	4760	6728	0000000194.png
(fce)	2419	556	2605	640	4760	6728	0000000194.png

El conjunto de datos usado para entrenar el modelo fue construido con la clase FunsDataset [8] del paquete layoutlm.data.funsd definido por el autor de LayoutLM [29]. Los parámetros de entrada son los archivos {train-test}.txt, {train-test}_box.txt, {train-test}_image.txt, cantidad máxima de palabras por documento (max_seg_length = 512), tokenizer microsoft/layoutlm-base-uncased [29], entidades y clases, tipo de datos (entrenamiento o test) y la respuesta es un objeto con 5 tensores. Cada tensor tiene un tamaño igual a la cantidad de facturas. Por último, con DataLoader[9] y los 5 tensores se construye el conjunto de datos final para entrenar y validar el modelo pre-entrenado de LayoutLM. La configuración de hiperparametros se corresponde con: optimizador AdamW (adaptive moment estimation) [11] con una taza de aprendizaje de 5e—5, 100 épocas. Cantidad de facturas para entrenamiento: 4118 – validación: 1030.

Tabla 5. Estadísticas de entrenamiento.

Loss	Precisión	Recall	F1	Accuracy
0.013035	0.98989	0.987773	0.988829	0.972029

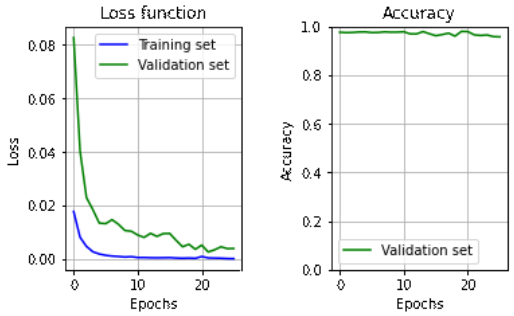

Fig. 6. Función de pérdida y precisión.

Como resultado final la predicción alcanza un ≈ 97% de precisión en 24 ciclos de entrenamiento, los resultados son presentados en tabla 5 y figura 6.

4 Test y Resultados

Para testear el modelo se desarrolló una aplicación Python con Flask[10] que permite subir un archivo de imagen o un archivo con extensión pdf. En ambos casos y según corresponda se ejecuta la tarea de preprocesamiento de acuerdo con 3.1 y 3.3 sin etiquetados. Como resultado se obtiene una predicción a nivel de cada token incluidos los separadores [CLS] [PAD] (ejemplo tabla 6), para comprobar visualmente las predicciones del modelo los bordes de cada palabra son remarcados en azul sin tener en cuenta las palabras con doble ## porque su ubicación está duplicada.

[8]https://pytorch.org/docs/stable/tensors.html

Tabla 6. Predicción a nivel de palabra.

Palabra	Input_id	Index	Clase
[CLS]	101	34	O
a	1037	1	I-INVOICE_TYPE_ID
cu	12731	18	B-CUIT_SELLER_ID
##it	4183	25	I-TOTAL_AMOUNT_ID
cu	12731	20	B-CUIT_CLIENT_ID
##it	4183	34	O
[PAD]	0	34	O

En la figura 7 se presenta un caso de prueba con todas las entidades de negocios halladas y en la figura 8 se presenta una sección ampliada del documento haciendo énfasis en el área que concentra una mayor cantidad de entidades. Para este último caso la anotación BIOES es eliminada con la intención de exponer la clasificación de entidad pura. En resumen, se puede considerar al modelo propuesto como aceptable considerando las estadísticas presentadas en tabla 5 y la comprobación visual sobre las facturas usadas en pruebas figura 7 y 8.

Fig. 7. Predicción en facturas Ejemplo 3.

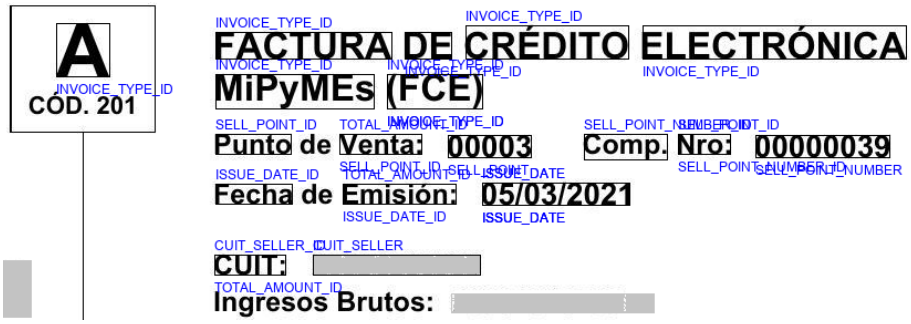

Fig. 8. Predicción en facturas Ejemplo 2.

Como ocurre en otros modelos de inteligencia artificial, errores son encontrados en la clasificación de entidades, la figura 9 exhibe una predicción incorrecta correspondiente al subtotal que no se encuentra etiquetado como entidad de interés.

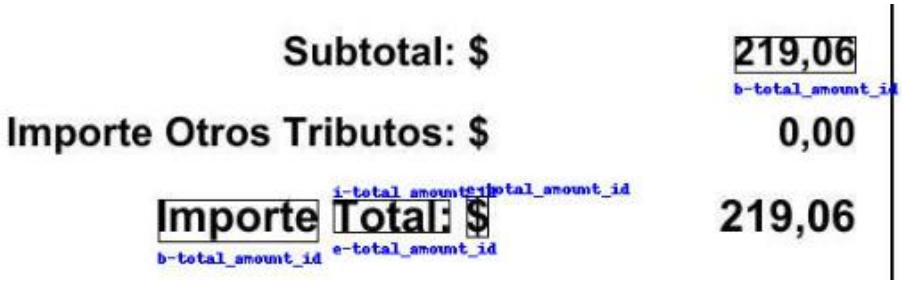

Fig. 9. Predicción incorrecta.

Ambiente de entrenamiento y pruebas:
Vendor GenuineIntel Google colab[11] Processor Intel(R) Xeon(R) CPU @ 2.30GHz, 2 Core Installed Physical Memory (RAM) 13.0 GB Hard Disc 80GB GPU NVIDIA-SMI 460.32.03 Driver Version: 460.32.03 CUDA Version: 11.2 Tesla K80 12GB RAM

5 Integración de información con fuentes heterogéneas

El procesamiento de cada documento con un modelo de inteligencia artificial (LayoutLM) retorna una predicción que puede ser jerarquizada según figura 10. Para este trabajo el nodo raíz es la factura propiamente dicha y sus nodos hijos son Tipo, Cuit (cliente-vendedor), Nro. Cae (Código de Autorización Electrónico) y otras entidades sin interés. Un segundo nivel tiene como raíz al Cuit del vendedor -quien emite la factura- y sus nodos hijos son Punto de venta, Fecha de emisión e Importe total facturado del documento o G=(V,E) donde V={invoice, nro cae, type, o, cuit_cliente, cuit_seller, total amount, sell point, issue date} y E={(invoice,nro cae), (invoice,type),(invoice,cuit_client),(invoice,cuit seller),(invoice,o),(cuit seller, sell point),(cuit seller, issue date),(cuit seller,total amount)}.

[11]https://colab.research.google.com/

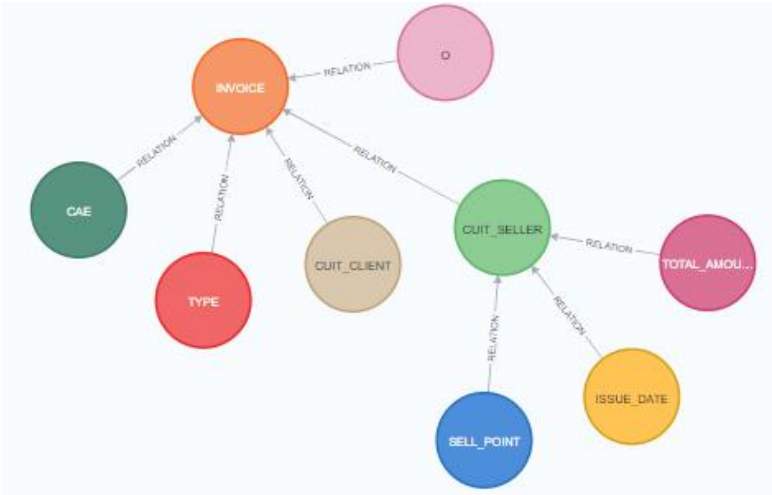

Fig. 10. Representación jerárquica de la información.

La estructura final de cada nodo a priori es desconocida porque depende de la información del documento procesado, de la librería OCR usada y del resultado de la predicción del modelo implementado, por esto es necesario ejecutar un análisis adicional con programación tradicional. Un motor de reglas como herramienta de análisis [12] nos permite enriquecer un modelo de datos heterogéneos, haciendo posible una amplia resolución de problemas de negocios [13, 14].

La configuración de esta herramienta consta de 4 pasos:

1 – Parametrizar los objetos del dominio desde mensajes json. Crear un Nivel (Level) para contener todo el mensaje json figura 11. Crear una entidad raíz (invoice) y un árbol de atributos con etiquetas(label) y un tipo de datos(data type). El objetivo de esta configuración es abstraer al analista de negocios a su dominio en tiempo de construcción de la regla, ver fig 11.

```
{
    "invoice": {
        "file" : { "label" : "factura.pdf", "value" : "SUDIDA...", "data_type" : "base64" },
        "type" : { "label" : "Tipo", "value" : "C", "data_type" : "string" },
        "others" : { "label" : "razon social", "value" : "XXXXXXXX", "data_type" : "string" },
        "cae" : { "label" : "CAE", "value" : "1234567", "data_type" : "string" },
        "seller" : {
            "cuit" : { "label" : "CUIT Vendedor", "value" : "30-xxxxxxxx-1", "data_type" : "string" },
            "sell_point" : { "label" : "Punto Vta", "value" : "0002", "data_type" : "string" },
            "issue_data" : { "label" : "Fecha Emision", "value" : "01/09/2022", "data_type" : "date" },
            "total_amount" : { "label" : "Importe Factura", "value" : "50000", "data_type" : "decimal" }
        },
        "client" : {
            "cuit" : { "label" : "CUIT Cliente", "value" : "30-xxxxxxxx-1", "data_type" : "string" }
        }
    }
}
```

Fig. 11. Transformación de objeto json a objeto del domino.

Para usar el input (fig. 11) se debe mapear por ejemplo a L(nivel)={invoice}, E(entidad)={invoice.cae.value}, TD(tipo dato)={json}.

2 – Construir los paquetes de reglas, cuya función es agrupar reglas.

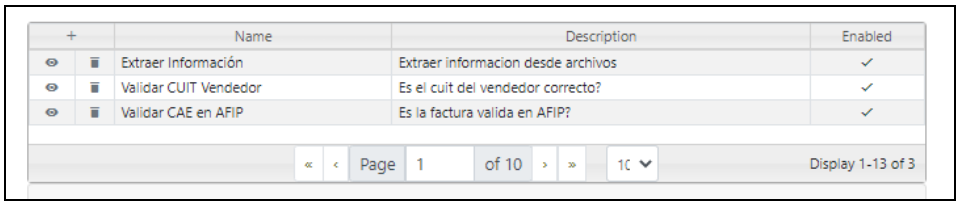

Fig. 12. Transformación de objeto json a objeto del domino.

La fig 12 presenta un paquete de regla para el caso de uso analizado en este artículo y compuesto de 3 reglas. Cada regla es ejecutada de modo secuencial, la respuesta de una regla es inyectada al contexto en tiempo de ejecución y disponible para consumir como información de entrada en la próxima regla [12].

3 – Construir las funciones aplicadas en cada regla.

Las funciones son componentes de software encapsulados y programados en el motor por el equipo de IT. Estas funciones ejecutan acciones como llamar a un servicio externo, validar condiciones con expresiones regulares, validaciones especificas, etc. Al momento de ejecutar la regla, el motor extrae la entidad y el valor del atributo para pasarlo por parámetro a la función parametrizada en la misma regla, fig 13.

```
Función Api.extraer.IA
curl --location --request POST 'localhost:9000/extraccion' \
--header 'Content-Type: application/json' \
--data-raw '{"data": {invoice.file.value}}

Función Api.validar_factura
curl --location --request POST 'localhost:9000/validarFactura' \
--header 'Content-Type: application/json' \
--data-raw '{"data": {invoice.cae.value}}
```

Fig. 13. Código de funciones.

4 – Construir las reglas.

Una regla es R(regla)={L(nivel)+E(entidad)+F(funcion)} donde F={ Api.extrae.IA}, ver fig 14.

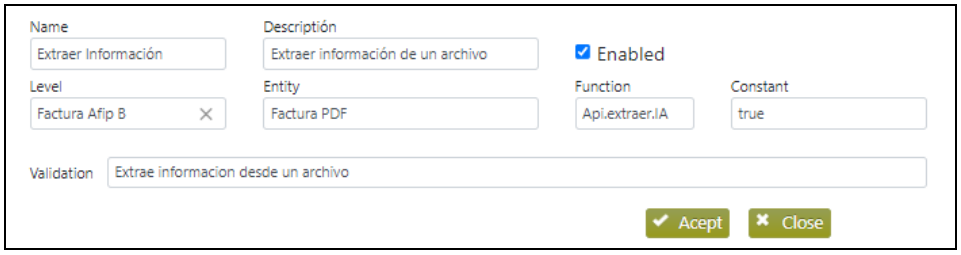

Fig. 14. Pantalla para crear reglas.

En nuestro caso de uso es tan importante el proceso de leer datos de un documento digital como válidar que dicho documento contenga información emitída por un ente público Argentino. Para cumplir con este objetivo dos valores CUIT-CAE son posteriormente validados contra una expresión regular y una base local en un servicio propio. El valor del CAE es consultado contra la entidad emisora, en este caso AFIP, quien da cuenta que la factura fue emitida de acuerdo a la reglamentacion pública.

Tabla 7. Ejecución de paquete de reglas.

Input	Regla	Resultado
{'invoice':{ 'file' : '/9j/4AAQSkZJ…' }	Extracción Información	{ 'invoice': { 'type' : 'C', 'cae' : '1234567', 'seller' : { 'cuit' : '30-xxxxxxxx-1', 'sell_point' : '0002', 'issue_data' : '01/09/2022', 'total_amount' : '50000' }, 'client' : { 'cuit' : '30-xxxxxxxx-1'}}}
{'invoice': {'seller': 'cuit': 30-xxxxxxxx-1'}	Valida CUIT Vendedor	{ 'isCuitOk' : true}
{'invoice': {'cae': '123456…'}	Valida CAE en AFIP	{ 'isCaeValido' : true}

La tabla 7 presenta la ejecución de un paquete de regla, el orden secuencial de ejecución comienza con la llamada a una regla con un objeto json como parametro de entrada, ejecutar la función asignada cuya respuesta es inyectada al contexto para la próxima ejecución.

Para el caso de uso detallado en este artículo, la información extraída de un archivo mediante el procesamiento con inteligencia artificial es reprocesada en un motor de reglas [12, 13, 14]. Este análisis posterior tiene como objetivo mejorar la calidad y robustecer la información para mejorar la toma de decisiones.

6 Conclusión y trabajos futuros

Presentamos la extracción de información de documentos argentinos agrupados en jerarquías de negocios para su posterior procesamiento en un motor de reglas. Un nuevo conjunto de datos fue desarrollado desde documentos digitalizados clasificado por extensión y diseño, se hizo una extracción de información con un OCR sumando ajustes para mejorar la calidad de los datos. El proceso de anotación de clases fue

desarrollado en una etapa automática y otra manual para finalmente adecuar el lote completo al formato esperado por el modelo LayoutLM. Como resultado final, las predicciones son agrupadas en una jerarquía de negocios ([FACTURA] - [VENDEDOR] - [CLIENTE]) expresada en formato json, donde cada par atributo-valor puede ser reutilizado en un motor de reglas para validar, analizar y/o incrementar la información mediante la ejecución de reglas encadenadas. El motor de reglas presentado esta orientado a servicios, con esta caractacterística es posible combinar la ejecución de modelos de inteligencia artificial con programación tradicional en cualquier orden, este enfoque es mas robusto que el uso por separado.

Futuras investigaciones incluyen agregar nuevos diseños como por ejemplo facturas de servicios de electricidad, internet, señales de televisión por cable y/o gas natural. Otro aspecto para investigar es el reconocimiento de imágenes dentro de imágenes para aquellas facturas generadas por emisores privados.

Referencias

1. Afip - Administración Federal de Ingresos Públicos Argentina. 2022. https://www.afip.gob.ar/ (accedido abril 09, 2022).
2. Bardelli, C., Rondinelli, A., Vecchio, R., & Figini, S. (2020). Automatic electronic invoice classification using machine learning models. Machine Learning and Knowledge Extraction, 2(4), 617-629.
3. Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document transformer. arXiv preprint arXiv:2004.05150.
4. Bertin Klein, Stevan Agne, and Andreas Dengel. 2004. Results of a study on invoice-reading systems in germany. In Simone Marinai and Andreas R. Dengel, editors, Document Analysis Systems VI. Springer Berlin Heidelberg, Berlin, Heidelberg, pages 451–462.
5. Biblioteca Afip - Modelo de Facturas http://biblioteca.afip.gob.ar/pdfp/rg_afip_0163_a001_v000.htm. (accedido abril 09, 2022).
6. Colacicchi, L. (2022). Comparison and fine-tuning of methods for Financial Sentiment Analysis.
7. Graliński, F., Stanisławek, T., Wróblewska, A., Lipiński, D., Kaliska, A., Rosalska, P., ... & Biecek, P. (2020). Kleister: A novel task for information extraction involving long documents with complex layout. arXiv preprint arXiv:2003.02356.
8. Guillaume Jaume, Hazim Kemal Ekenel, and Jean-Philippe Thiran. 2019. FUNSD: A Dataset for Form Understanding in Noisy Scanned Documents. 2019 International Conference on Document Analysis and Recognition Workshops (ICDARW) 2 (2019), 1–6.
9. Han, X., Zhang, Z., Ding, N., Gu, Y., Liu, X., Huo, Y., ... & Zhu, J. (2021). Pre-trained models: Past, present and future. AI Open, 2, 225-250.
10. Hugging Face. 2022. Transformers. https://github.com/huggingface/transformers (accessed November 27, 2022).
11. Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.
12. Maciel, Marcos (2020). Motor de Reglas desacoplado orientado a formato JavaScript Object Notation. XXVI Congreso Argentino de Ciencias de la Computación, CACIC Buenos Aires, Argentina. Páginas 489-498 ISBN: 0201633612
13. Marcos Maciel & Claudia Pons (2021a). Implementación Técnica de una Arquitectura Orientada a Integrar Conocimiento Externo Heterogéneo en Motor de Reglas. Congreso Argentino de Ciencias de la Computación CACIC. ISBN 978 987 633 574 4. Universidad Nacional de Salta, Argentina. Páginas 583-593
14. Marcos Maciel & Claudia Pons (2021b). Orquestación de Reglas para Integrar Conocimiento Heterogéneo. 9no.Congreso Nacional de Ingeniería Informática / Sistemas de Información CoNaIISI 2021. ISBN 978-950-42-0213-4 UTN, Facultad Regional Mendoza, Argentina. Páginas 413-418

15. D. Ming, J. Liu, and J. Tian, “Research on chinese financial invoice recognition technology,” Pattern recognition letters, vol. 24, no. 1, pp. 489–497, 2003.
16. Liu, W., Wan, B., & Zhang, Y. (2016). Unstructured Document Recognition on Business Invoice CS 229 : Machine Learning.
17. Motahari, H., Duy, N., Bennett, P., Bedrax-Weiss, T.: A report on the rst workshop on document intelligence (di) at neurips 2019. ACM SIGKDD Explorations Newsletter 22(2), 8{11 (2021) Ray Smith. 2020. Tesseract Open Source OCR Engine.
18. Neyshabur, B., Sedghi, H., & Zhang, C. (2020). What is being transferred in transfer learning?. Advances in neural information processing systems, 33, 512-523.
19. Clement Sage. Deep learning for information extraction from business documents. Machine Learning [stat.ML]. Université de Lyon, 2021. English. NNT: 2021LYSE1172. tel-03521607
20. Clément Sage, Thibault Douzon, Alex Aussem, Véronique Eglin, Haytham Elghazel, et al.. Data-Efficient Information Extraction from Documents with Pre-Trained Language Models. ICDAR 2021 Workshop on Document Images and Language, Sep 2021, Lausanne, Switzerland. hal-03267497
21. Stanisławek, T., Graliński, F., Wróblewska, A., Lipiński, D., Kaliska, A., Rosalska, P., ... & Biecek, P. (2021, September). Kleister: key information extraction datasets involving long documents with complex layouts. In International Conference on Document Analysis and Recognition (pp. 564-579). Springer, Cham.
22. Sun, C., Qiu, X., Xu, Y., & Huang, X. (2019, October). How to fine-tune bert for text classification?. In China national conference on Chinese computational linguistics (pp. 194-206). Springer, Cham.
23. Szegedi, G., Veres, D.B., Lendák, I., & Horváth, T. (2020). Context-based Information Classification on Hungarian Invoices. ITAT.
24. Tarawneh, Ahmad & Hassanat, Ahmad & Chetverikov, Dmitry & Lendak, Imre & Verma, Chaman. (2019). Invoice Classification Using Deep Features and Machine Learning Techniques. 10.1109/JEEIT.2019.8717504.
25. von Rueden, Laura & Houben, Sebastian & Cvejoski, Kostadin & Bauckhage, Christian & Piatkowski, Nico. (2022). Informed Pre-Training on Prior Knowledge.
26. von Rueden, Laura & Mayer, Sebastian & Beckh, Katharina & Georgiev, Bogdan & Giesselbach, Sven & Heese, Raoul & Kirsch, Birgit & Walczak, Michal & Pfrommer, Julius & Pick, Annika & Ramamurthy, Rajkumar & Garcke, Jochen & Bauckhage, Christian & Schuecker, Jannis. (2021). Informed Machine Learning - A Taxonomy and Survey of Integrating Prior Knowledge into Learning Systems. IEEE Transactions on Knowledge and Data Engineering. PP. 1-1. 10.1109/TKDE.2021.3079836.
27. Xavier Holt and Andrew Chisholm. 2018. Extracting structured data from invoices. In Proceedings of the Australasian Language Technology Association Workshop 2018, pages 53–59, Dunedin, New Zealand, December.
28. Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, GuoxinWang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, et al. 2020a. Layoutlmv2: Multi-modal pre-training for visually-rich document understanding. arXiv preprint arXiv:2012.14740.
29. Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020b. Layoutlm: Pretraining of text and layout for document image understanding. In Proceedings of the 26th ACMSIGKDD International Conference on KnowledgeDiscovery & Data Mining, pages 1192–1200.
30. Ha, H. T., Neveˇˇrilova, Z., Hor ´ ak, A., et al. (2018). Recog- ´ nition of OCR Invoice Metadata Block Types. In Text, Speech, and Dialogue. TSD 2018, pages 304– 312. Springer, Cham
31. Lample, Guillaume & Ballesteros, Miguel & Subramanian, Sandeep & Kawakami, Kazuya & Dyer, Chris. (2016). Neural Architectures for Named Entity Recognition. 260-270. 10.18653/v1/N16-1030.