Comparativa de modelos para el reconocimiento de estructuras de datos tabulares: Un enfoque desde el aprendizaje profundo

Model Comparison for Table Structure Recognition: A Deep Learning Approach

JOSE MIGUEL GARZÓN VARGAS
jmgarzonv1@eafit.edu.co

Trabajo de grado para optar al título de Magíster en Ciencia de Datos y Analítica

Director:

Juan David Martínez Vargas
jdmartinev@eafit.edu.co

Codirectora:

Lina María Sepúlveda Cano
lmsepulvec@eafit.edu.co

UNIVERSIDAD EAFIT

ESCUELA DE CIENCIAS APLICADAS E INGENIERÍA

MAESTRÍA EN CIENCIAS DE LOS DATOS Y LA ANALÍTICA

MEDELLÍN

2024

Resumen

En el ámbito de la digitalización documental, las tablas representan una herramienta fundamental para la organización y transmisión efectiva de información en diversas industrias. Es particularmente relevante en documentos comerciales como facturas y órdenes de compra, donde los detalles asociados a las transacciones suelen estar dispuestos en estructuras que siguen una lógica similar a la de las tablas. Estas estructuras, si bien están diseñadas para la comprensión intuitiva por parte de los humanos, presentan un reto considerable para los sistemas de extracción de información automatizados.

Aunque existen múltiples esfuerzos para abordar este problema, sus resultados no son fácilmente comparables por la falta de consistencia en los conjuntos de datos utilizados para el entrenamiento y muestra de resultados, y por los propios sesgos en la información que presentan los conjuntos de datos disponibles. En este contexto, este proyecto se enfoca en realizar una evaluación comparativa del desempeño de modelos de aprendizaje profundo especializados en el reconocimiento de la estructura de datos tabulares utilizando la metodología CRISP-DM.

Se presenta entonces una comparativa de dos modelos relevantes en el estado del arte: *Table Transformer* y *Unitable*, describiendo sus características y evaluando la calidad de sus inferencias con el conjunto de datos *SynthTabNet*, reconocido por su diversidad y relevancia. Se estandariza la representación de la estructura de la tabla que infiere cada sistema al formato HTML utilizado en SynthTabNet, y se emplea la similitud basada en la distancia de edición de árboles como la métrica de comparación entre las inferencias de los modelos y los datos anotados. Se evidenció la superioridad de *Unitable* sobre *Table Transformer* en la representación de tablas complejas.

El proyecto desarrollado para estandarizar la comparación de estos dos modelos y sus resultados puede ser consultado en el repositorio público de este proyecto https://github.com/JmGarzon/TSR_model_comparison.

Palabras clave: Reconocimiento de estructura de datos tabulares; Modelos de aprendizaje profundo; Similitud basada en la distancia de edición de árboles.

Descripción del Proyecto

1 Planteamiento del Problema

Las tablas suelen ser medios para representar y comunicar información estructurada en diversos escenarios. En el contexto de la digitalización, el reconocimiento automático de la estructura de una tabla ha sido un tema importante dentro de la investigación de la comprensión automática de documentos (Lin et al., 2022).

El reconocimiento de estructuras de tablas busca determinar las estructuras celulares de tablas a partir de imágenes, al extraer las coordenadas de las celdas, filas y columnas que recogen la información. No es una tarea trivial dado que las tablas pueden tener estructuras complejas, utilizando variados estilos para representar información diversa (Lin et al., 2022).

Múltiples esfuerzos se han hecho para reconocer estructuras tabulares desde diversos enfoques como los motores de reglas, modelos supervisados, semi supervisados y no supervisados; y diversos modelos de aprendizaje profundo (Kashinath et al., 2022). Sin embargo, no son fácilmente comparables por la falta de consistencia en los conjuntos de datos utilizados para el entrenamiento y muestra de resultados, y por los propios sesgos en la información que presentan los conjuntos de datos disponibles. Este trabajo busca implementar y comparar modelos de aprendizaje profundo preentrenados del estado del arte, para la detección de estructura de tablas; aprovechando los recientes esfuerzos que se han hecho en la construcción de conjuntos de datos diversos para su comparación (Nassar et al., 2022).

Es importante aclarar que una etapa previa a la detección de la estructura de la tabla es la detección de la tabla como entidad. La detección de tablas es un problema ampliamente estudiado dado su parecido con los modelos de detección de objetos; por lo que cuenta con modelos que actualmente generan muy buenos resultados (Fernandes et al., 2023) y no son objeto de estudio de este trabajo.

2 Justificación

En el marco de la transformación digital, la extracción de información de documentos es una tarea común pero relevante en ámbitos financieros, manufactureros y de comercio. Dado que la extracción manual de información es un proceso sumamente tedioso y demorado, cobra relevancia el desarrollo de sistemas de extracción de información automáticos que reduzcan el tiempo y el esfuerzo asociado a la labor (Riba et al., 2019).

Un reto particular en la extracción de información de documentos comerciales es el reconocimiento de los elementos involucrados en la transacción, dado que no necesariamente se encuentran demarcadas o siguen un patrón generalizable, por ejemplo, al separar su información en múltiples líneas de texto (Holeček et al., 2019). Sin embargo, es común encontrarlos dentro de estructuras semejantes a tablas, en donde su información de interés, como el identificador, su descripción, o precio; se encuentran agrupadas en una misma fila separada en columnas (Riba et al., 2019).

Es en este punto donde cobra relevancia el reconocimiento de estructuras tabulares en imágenes (Šimsa et al., 2023), en donde la correcta identificación de lo que corresponde a una fila y columna de una determinada tabla, permite asociar la información relacionada sin necesidad de intervención humana. Cumpliendo una etapa importante del posterior reconocimiento de los componentes de los campos de interés de los elementos involucrados en la transacción.

3 Objetivos

3.1 Objetivo general:

Desarrollar una estrategia de comparación de modelos de detección de estructuras de tablas utilizando técnicas de aprendizaje profundo.

3.2 Objetivos específicos:

- Seleccionar y adaptar modelos de detección de estructuras de tablas basados en aprendizaje profundo disponibles en el estado del arte.
- Implementar un entorno de desarrollo de experimentos para comparar el rendimiento de los diferentes modelos seleccionados.
- Identificar métricas de evaluación que reflejen con precisión el rendimiento de los modelos en la detección de estructuras de tablas.
- Documentar y presentar el desempeño de los modelos bajo la métrica seleccionada, las fortalezas y debilidades de cada enfoque y las diversas conclusiones alcanzadas.

4 Marco teórico y estado del arte

Si bien la idea de una tabla es concebida intuitivamente para la mayoría de las personas, una definición formal no es una tarea sencilla dada la diversidad de formas y contextos en los que las tablas son construidas y utilizadas. Una tabla es un medio prevalente para la representación y comunicación de información estructurada (Coüasnon & Lemaitre, 2014). Pueden contener palabras, números, formulas, e incluso gráficos (Embley et al., 2006).

Silva et al. (2006) proponen que una tabla es una representación gráfica en forma de cuadrícula de una matriz $M_{i,j}$ que cumpla con:

- Cada elemento i, j de la matriz es atómico;
- Hay pistas visuales lineales, es decir, los elementos de cada fila i (columna j) de la matriz tienden a estar alineados horizontalmente (verticalmente);
- Las pistas visuales lineales describen conexiones lógicas, es decir, algunos de los elementos de cada fila i (columna j) están vinculados en su significado entre sí y no son independientes; como tal, cada fila (columna) de una tabla es inquebrantable, en el sentido de que si dos elementos en una fila dada (columna) intercambian lugares, entonces las dos filas (columnas) correspondientes deben hacerlo también;
- El arte de líneas eventual no agrega significado que no esté presente en el posicionamiento relativo de las celdas en la tabla.

Ahora bien, como lo mencionan Göbel et al (2012), el problema del entendimiento automático de tablas consiste en tres tareas:

- Detección de tabla: Encontrar regiones de un documento con contenidos tabulares.
- Detección de la estructura de la tabla: Reconstrucción de la estructura celular de una tabla.
- Interpretación de la tabla: Descubrimiento del sentido de la estructura tabular, incluyendo:
    - Análisis funcional: Determina la función de las celdas y sus relaciones lógicas.
    - Interpretación semántica: Entendimiento de la semántica de una tabla en términos de las entidades que se representan en ella, sus atributos, e interrelaciones.

Uno de los primeros intentos de comparar el desempeño de modelos para el entendimiento automático de tablas fue presentado por Gobel et al. (2013), en el contexto de la Conferencia Internacional sobre Análisis y Reconocimiento de Documentos (ICDAR, por sus siglas en inglés). Se generó un conjunto de datos que comprendía 156 tablas de documentos en formato PDF de varios sitios web gubernamentales de la unión europea que aún se utiliza como punto de comparación para nuevos modelos (Fernandes et al., 2023). Entre las soluciones a comparar se encontraban productos comerciales como el *ABBYY FineReader 11.0 Corporate Edition*, el *Adobe Acrobat XI Pro*, el *OmniPage 18 Professional*, y el *Nitro Pro-8* que mostraron en general un mejor desempeño que las soluciones académicas (Gobel et al., 2013). Entre los retos más

relevantes encontrados en esta comparativa estaban también las estructuras complejas en los encabezados de las tablas y las tablas pequeñas. Tablas con menos de cinco filas suponían un reto frecuente dado que la mayoría de los algoritmos partían de la detección de una semilla base, que pasaban por alto este tipo de tablas (Gobel et al., 2013).

Luego, el marco de la ICDAR 2019, Gao et al. (2019) proponen una nueva competencia para la detección y el reconocimiento de tablas en la que recopilaron una gran variedad de tablas como tablas dibujadas a mano de libros contables, listas de la bolsa de valores, horarios de trenes, libros de registros, listas de prisioneros, impresiones de libros, mediciones de producción, entre otras. La competencia propone varias líneas de trabajo, la línea A para la comparación de métodos de detección de tablas y la línea B para el reconocimiento de tablas. La línea B a su vez cuenta con dos vertientes, la línea B.1 buscaba comparar métodos de reconocimiento de estructuras tabulares, y la línea B.2 buscaba comparar métodos integrales que ejecutaran tanto la detección de la tabla como el reconocimiento de su estructura. Para los efectos de este trabajo, se discutirán los datos y resultados de la línea B.1.

El conjunto de datos utilizado para la línea B.1 consiste en 750 tablas cuyas anotaciones comprenden tanto las entidades de la tabla en sí, como de sus celdas individuales estructuradas en formato XML. El método con mejor desempeño para esta categoría fue el *NLPR PAL* de Xiao-Hui Li, Fei Yin y Cheng-Lin Liu. Este método, basado en el etiquetado de componentes conectados, alcanzó un F1-score ponderado del 48%, indicando que hay amplias oportunidades de mejora para este tipo de soluciones (Gao et al., 2019).

Ahora bien, gracias a los avances en las unidades de cómputo y al creciente número de conjuntos de datos de alto volumen que han venido siendo puestos a disposición de la comunidad científica; los modelos basados en aprendizaje profundo han podido sobrepasar el desempeño de los métodos tradicionales para este tipo de tareas (Hashmi et al., 2021).

Hashmi et al. (2021) presenta una revisión de los principales métodos de aprendizaje profundo utilizados para el reconocimiento de la estructura de datos tabulares, los cuales se describen a continuación.

a) **Redes Neuronales Convolucionales (CNN, por sus siglas en inglés):** Yamashita et al. (2018) explica que las CNN son un tipo de modelo de aprendizaje profundo diseñado para procesar datos con un patrón de cuadrícula, como imágenes. Están inspiradas en la organización del córtex visual de los animales y están diseñadas para aprender de manera automática y adaptativa jerarquías espaciales de características, desde patrones de bajo nivel hasta patrones de alto nivel. Una CNN se compone típicamente de tres tipos de capas: convolución, agrupación y completamente conectadas. Las capas de convolución y agrupación realizan la extracción de características, mientras que la capa completamente conectada mapea las características extraídas hacia una salida final, como una clasificación (Yamashita et al., 2018).

Una capa de convolución juega un papel clave en una CNN y está compuesta por una serie de operaciones matemáticas, incluida la convolución, que es un tipo especializado de operación lineal. En imágenes digitales, los valores de píxeles se almacenan en una cuadrícula bidimensional (2D), es decir, un conjunto de números. En cada posición de la imagen se aplica un pequeño conjunto de parámetros llamado *kernel*, que es un extractor de características optimizable. Este *kernel* actúa como un extractor de características, permitiendo a la CNN detectar patrones específicos como bordes, texturas y formas en la imagen. Esto hace que las CNN sean altamente eficientes para el procesamiento de imágenes, ya que una característica puede aparecer en cualquier lugar de la imagen. A medida que una capa alimenta su salida a la siguiente capa, las características extraídas pueden volverse progresivamente más complejas y jerárquicas (Yamashita et al., 2018).

El proceso de optimización de parámetros como los *kernels* se denomina entrenamiento, que se realiza para minimizar la diferencia entre las salidas y las etiquetas de verdad mediante un algoritmo de optimización llamado retropropagación y descenso de gradiente, entre otros (Yamashita et al., 2018).

La arquitectura de una CNN está compuesta por varios componentes fundamentales, como capas de convolución, capas de agrupación y capas completamente conectadas. Una estructura típica consiste en repeticiones de una serie de capas de convolución seguidas de una capa de agrupación, seguidas a su vez por una o más capas completamente conectadas (Yamashita et al., 2018), como se muestra en la Figura 1.

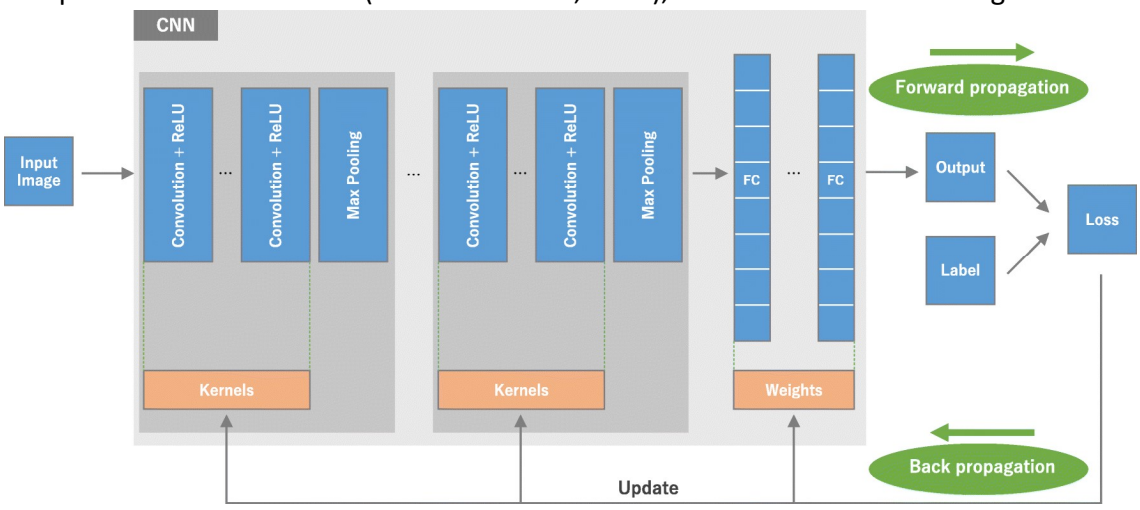

Figura 1. Arquitectura típica de una CNN y su proceso de entrenamiento (Yamashita et al., 2018).

Como lo menciona Li et al. (2022), un par de técnicas utilizadas para mejorar la capacidad de una CNN para capturar y procesar características de datos complejos, especialmente en el procesamiento de imágenes son la **convolución dilatada y deformada**.

La convolución dilatada se destaca por su capacidad para expandir el campo receptivo de los núcleos de convolución sin aumentar exponencialmente el número de parámetros. Al insertar valores vacíos entre los puntos del núcleo de convolución, esta técnica permite la captura de características más extensas en los datos de entrada, reduciendo la pérdida de información en los bordes (Li et al., 2022). Una comparativa entre un *kernel* tradicional y un *kernel* dilatado puede verse en la Figura 2.

Por otro lado, la convolución deformada aborda la limitación de las estructuras geométricas fijas en las redes neuronales, haciendo que los modelos sean más adaptables a transformaciones geométricas. Introduce dos módulos clave: el módulo de convolución deformada, que se centra en áreas específicas de interés mediante el aprendizaje de desplazamientos y su aplicación al núcleo de convolución, permitiendo adaptaciones dinámicas a escalas y transformaciones, y el módulo de *Pooling* de Región de Interés Deformado, que utiliza una capa completamente conectada para aprender y aplicar desplazamientos dinámicos en el proceso de *pooling* (Li et al., 2022).
Estas técnicas son normalmente utilizadas para mejorar la adaptabilidad de las CNN a objetos con formas irregulares o cambios en posición y orientación, sin depender excesivamente de técnicas de aumento de datos o conjuntos de entrenamiento adicionales (Li et al., 2022). Una comparativa entre un *kernel* tradicional y un *kernel* deformado puede verse en la Figura 3.

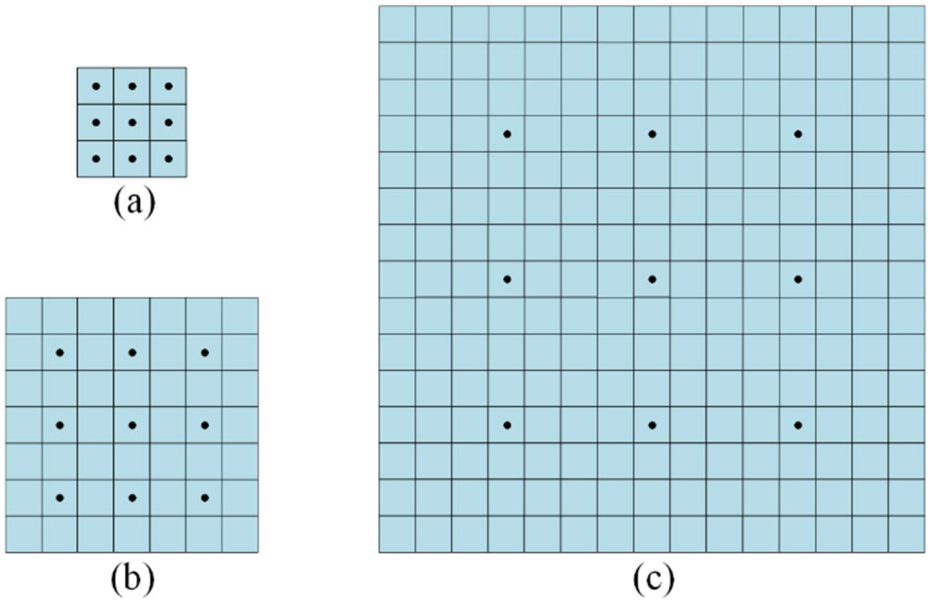

Figura 2. Comparación entre un *kernel* de convolución general y *kernels* de convolución dilatados. (a) *Kernel* general 3x3. (b) *Kernel* dilatado de índice 2. (c) *Kernel* dilatado de índice 4 (Li et al., 2022).

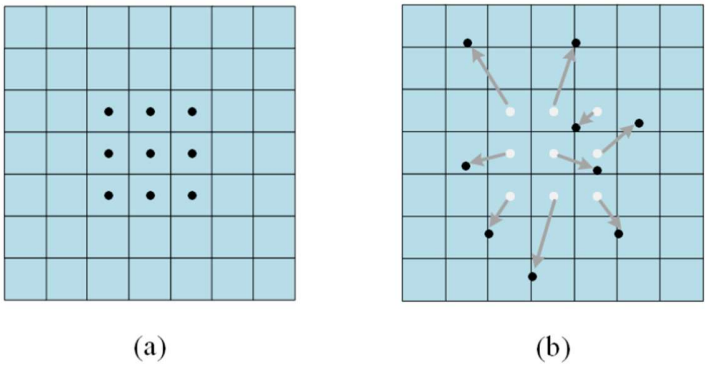

Figura 3. Comparación entre un *kernel* de convolución general y un *kernel* de convolución deformado. (a) *Kernel* de convolución general de 3 × 3. (b) *Kernel* de convolución deformado de 3 × 3 (Li et al., 2022).

b) **Redes Neuronales Completamente Convolucionales (FCN, por sus siglas en inglés):** Shelhamer et al. (2016) presenta las FCN como un tipo de arquitectura de red neuronal diseñada específicamente para tareas de segmentación semántica. A diferencia de las Redes Neuronales Convolucionales tradicionales, que están diseñadas para tareas de clasificación de imágenes, las FCN pueden producir una predicción densa píxel a píxel para una imagen de entrada. Esto significa que, en lugar de producir una sola etiqueta para toda la imagen, las FCN producen una etiqueta para cada píxel de la imagen, lo que es útil para tareas como la detección de objetos y la segmentación de imágenes (Shelhamer et al., 2016).

   La principal diferencia entre las FCN y las CNN es que las FCN reemplazan las capas completamente conectadas de una CNN con capas convolucionales. Esto permite que la red tome imágenes de entrada de cualquier tamaño y produzca mapas de salida del mismo tamaño, lo que es necesario para tareas de segmentación semántica(Shelhamer et al., 2016).

   Además, las FCN a menudo utilizan conexiones de salto (*skip connections*) para combinar características de diferentes capas de la red, lo que ayuda a preservar la información espacial y mejorar la precisión de la segmentación. La arquitectura de salto fusiona la jerarquía de características para combinar información profunda, gruesa y semántica con información superficial, fina y de apariencia (Shelhamer et al., 2016). Se ha demostrado que este enfoque supera los mejores resultados anteriores sin necesidad de maquinaria adicional, siendo efectivas para una variedad de tareas de segmentación semántica, incluyendo la detección de objetos y el análisis de escenas (Shelhamer et al., 2016).

c) **Redes Neuronales basadas en grafos (GNN, por sus siglas en inglés):** Zhou et al. (2018) define a las GNN como modelos especializados para procesar datos en forma de grafos, partiendo de que los grafos son herramientas poderosas y versátiles para representar

sistemas complejos y sus relaciones. Su flexibilidad para modelar conexiones entre elementos y capturar estructuras no lineales los hace ideales para situaciones con relaciones intrincadas (Zhou et al., 2018).

Su punto fuerte radica en su habilidad para captar relaciones y dependencias complejas dentro de los grafos, logrando esto a través de un proceso conocido como propagación de mensajes. Esta técnica les permite recopilar información de nodos cercanos en diferentes niveles, lo que a su vez proporciona una comprensión profunda de la estructura subyacente del grafo (Zhou et al., 2018).

Una de las ventajas más notables de las GNN es su capacidad para manejar estructuras de datos no estándar, superando así las limitaciones de las redes neuronales convencionales que están diseñadas para trabajar con cuadrículas, como las CNN; ofreciendo así una forma de representar de manera natural sistemas con conexiones intrincadas. Este enfoque es especialmente útil en situaciones donde las relaciones entre elementos no siguen un patrón regular (Zhou et al., 2018).

Las GNN adaptan principios provenientes de las CNN para procesar datos de grafos. Esto implica la incorporación de conceptos fundamentales como la conexión local, pesos compartidos y estructuras de múltiples capas, todos ellos adaptados para trabajar con las particularidades de los grafos (Zhou et al., 2018).

En lo que respecta a la segmentación semántica, las regiones en las imágenes a menudo no siguen una estructura de cuadrícula y requieren información no local, lo que lleva al fracaso de las CNN tradicionales, que están diseñadas para operar en cuadrículas estándar. Para abordar este desafío, se han realizado diversos trabajos que emplean datos con estructura de grafo y GNN (Zhou et al., 2018).

d) **Redes Neuronales Recurrentes (RNN, por sus siglas en inglés):** Como lo indica Staudemeyer & Morris (2019), una RNN se destaca como una categoría especializada dentro de las redes neuronales artificiales, diseñada específicamente para el análisis de datos secuenciales o dinámicos. A diferencia de las redes neuronales convencionales de propagación hacia adelante, las RNN cuentan con conexiones recurrentes que les permiten incorporar información de pasos temporales anteriores. Esta característica arquitectónica dota a las RNN de una capacidad única para sobresalir en tareas caracterizadas por dependencias temporales, donde el orden de los puntos de datos es significativo (Staudemeyer & Morris, 2019). Los nodos dentro de una RNN forman colectivamente una red interconectada, facilitando la transmisión fluida de información de un paso temporal al siguiente (Staudemeyer & Morris, 2019).

   En el panorama evolutivo de las arquitecturas de redes neuronales, las Redes de Gran Memoria de Corto Plazo (LSTM, por sus siglas en inglés) surgen como un avance fundamental más allá de las RNN convencionales. Las LSTM se presentan como una respuesta estratégica a las limitaciones identificadas en las RNN tradicionales, ofreciendo capacidades mejoradas para procesar secuencias más extensas al tiempo

que mitigan problemas asociados con señales que desaparecen o se disparan (Staudemeyer & Morris, 2019).

Modelos basados en RNN han sido utilizados para una gran variedad de tareas como el reconocimiento de escritura y habla, la identificación de emociones, sistemas de traducción, la generación de texto y música, entre muchos otros (Staudemeyer & Morris, 2019).

Para la tarea de procesamiento de imágenes, se han utilizado modelos más sofisticados como las Redes de Gran Memoria de Corto Plazo Bidireccionales (BLSTM, por sus siglas en inglés) (Hashmi et al., 2021). Como lo indica Staudemeyer & Morris (2019), las BLSTM constituyen una extensión de la arquitectura convencional del LSTM. En el contexto de la BLSTM, la secuencia de entrada se presenta tanto en dirección hacia adelante como hacia atrás a dos redes LSTM separadas, ambas conectadas a la misma capa de salida. Este enfoque bidireccional permite analizar tanto el pasado como el futuro de un punto dado en la secuencia, demostrando ventajas arquitectónicas, especialmente en la clasificación de fonemas (Staudemeyer & Morris, 2019).

e) **Red Neuronal Convolucional basada en Regiones (R-CNN, por sus siglas en inglés):** Bharati Pujaand Pramanik (2020) presenta una revisión de las R-CNN y los modelos que las usaron como base. Parte definiendo las R-CNN como una arquitectura específica diseñada para la detección de objetos en imágenes. La R-CNN original opera en tres pasos: la generación de propuestas de región mediante un algoritmo denominado *selective search*, la aplicación de una red neuronal convolucional a cada propuesta de región, la clasificación de las regiones utilizando una máquina de vector soporte para la identificación del objeto, y un regresor lineal para refinar la ubicación de la caja delimitadora (Bharati Pujaand Pramanik, 2020).
Sin embargo, la R-CNN presenta limitaciones notables, como el extenso tiempo necesario para entrenar la red al clasificar alrededor de 2000 propuestas de región por imagen, la incapacidad de realizar detección en tiempo real, y la complejidad del proceso de entrenamiento debido a la necesidad de entrenar tres modelos separados (Bharati Pujaand Pramanik, 2020).
Para superar estas limitaciones, se han propuesto diversas mejoras a lo largo del tiempo. *Fast* R-CNN elimina la generación separada de propuestas de región al alimentar directamente la imagen a la red neuronal convolucional y realiza el entrenamiento simultáneo de los componentes clave. *Faster* R-CNN introduce una Red de Propuesta de Región (RPN) para acelerar la generación de propuestas. R-FCN, por su parte, busca maximizar el uso compartido de cálculos computacionales para lograr una mayor eficiencia y velocidad en la detección de objetos. SSD realiza la detección de objetos en una sola pasada, aumentando la eficiencia. YOLO adopta un enfoque de cuadrícula para predicciones rápidas, aunque puede tener dificultades con objetos pequeños o de formas irregulares (Bharati Pujaand Pramanik, 2020).

La evolución culmina en *Mask* R-CNN, que extiende *Faster* R-CNN para incluir la segmentación de instancias, utilizando una red totalmente conectada (FCN) para la segmentación semántica y mejorando la precisión mediante la introducción de *RoIAlign* para la segmentación precisa a nivel de píxeles. Estas mejoras buscan abordar las limitaciones de la R-CNN original, mejorando la eficiencia, velocidad y precisión en la detección y segmentación de objetos en imágenes.

Por otro lado, autores como Smock et al. (2021) y Peng et al. (2024) utilizan *Transformers* como herramienta principal para el desarrollo de modelos de reconocimiento de tablas. Los *Transformers* fueron introducidos por Vaswani et al. (2017) como un nuevo bloque de construcción basado en mecanismos de atención para modelos de aprendizaje automático. Como lo expone Carion et al. (2020), los mecanismos de atención son capas de redes neuronales que agregan información de toda la secuencia de entrada. Los *Transformers* introdujeron capas de autoatención que escanean cada elemento de una secuencia y lo actualizan agregando información de toda la secuencia. Una de las principales ventajas de los modelos basados en atención es su capacidad de realizar cálculos globales y su memoria perfecta, lo que los hace más adecuados que las redes neuronales recurrentes para secuencias largas. Actualmente, los *Transformers* están siendo utilizados en diversos problemas de procesamiento de lenguaje natural, procesamiento de voz y visión por computadora (Carion et al., 2020).

Ahora bien, han sido publicados múltiples modelos que buscan solucionar el problema de la detección de estructura de tablas desde diferentes enfoques. Sin embargo, una condición necesaria para hacer comparaciones efectivas es la replicabilidad de los modelos. La Tabla 1, reúne entonces publicaciones recientes de código abierto.

Tabla 1. Modelos publicados en el estado del arte.

Referencia	**Año de publicación**	**Nombre**
(Xue et al., 2019)	2019	*Res2TIM*
(Prasad et al., 2020)	2020	*Cascade TabNet*
(Fischer et al., 2021)	2021	*Multi-Type-TD-TSR*
(Xue et al., 2021)	2021	*TGRNet*
(Smock et al., 2021)	2021	*TATR*
(Lee et al., 2022)	2022	*Graph – based -TSR*
(Peng et al., 2024)	2024	*UniTable*

Otra pieza fundamental para la implementación y comparación de los modelos son los datos. En este punto resalta el trabajo de Nassar et al. (2022), en donde presentan *SynthTabNet*, un conjunto de datos sintéticos de tablas etiquetadas que busca solucionar los problemas de los conjuntos de datos tradicionales para la detección de estructura de tablas. Se ha demostrado que otros conjuntos de datos, como *PubTabNet*, *FinTabNet* y *TableBank*, presentan varias limitaciones, como distribuciones sesgadas hacia estructuras más simples, apariencias limitadas

y contenido restringido a ciertos dominios (Nassar et al., 2022). *SynthTabNet* aborda estas limitaciones al proporcionar una amplia gama de tamaños de tablas, combinaciones diversas de extensiones de filas y columnas, estilos de apariencia específicos de varios dominios y cajas delimitadoras para todas las celdas de las tablas, incluso las vacías (Nassar et al., 2022). *SynthTabNet* está organizado en 4 partes de 150,000 tablas cada una (600,000 en total), con divisiones de Entrenamiento, Prueba y Validación (80%, 10%, 10%). Las tablas se entregan como imágenes *png* y las anotaciones están en formato *jsonl* (Nassar et al., 2022).

5 Metodología

CRISP-DM, del inglés *"CRoss Industry Standard Process for Data Mining"*, es la metodología estándar para la aplicación de proyectos de minería y ciencia de datos que consiste en seis etapas iterativas desde el entendimiento del negocio hasta el despliegue (Schröer et al., 2021). A continuación, se describe brevemente en qué consiste cada etapa de esta metodología y su relación con el proyecto.

- **Entendimiento del negocio:** La situación problemática debe evaluarse para obtener una visión general de los recursos disponibles y necesarios. La definición del objetivo de la minería de datos es uno de los aspectos más importantes en esta fase (Wirth & Hipp, 2000).

  El proceso de definición de objetivos de este proyecto se consolida con la presentación de esta propuesta en donde, luego de entender las características del problema de detección de la estructura de tablas, su relevancia, y la dificultad de comparar los diferentes modelos disponibles en el estado del arte; se hace una revisión bibliográfica que permite identificar los elementos conceptuales que fundamentan los modelos a estudiar, los recursos disponibles y los recientes avances que se han hecho desde diferentes enfoques. Es así como el objetivo de este define como:

  > **Desarrollar una estrategia de comparación de modelos de detección de estructuras de tablas utilizando técnicas de aprendizaje profundo.**

- **Entendimiento de los datos:** Recopilar datos de fuentes de datos, explorarlos, describirlos y verificar la calidad de los datos son tareas esenciales en esta fase (Wirth & Hipp, 2000).

  El conjunto de datos a utilizar para el desarrollo de este proyecto es *SynthTabNet*, propuesto por Nassar et al. (2022). Presentan un total de cuatro conjuntos de datos sintéticos, cada uno integrado por 150,000 ejemplos. La generación del texto de las tablas se basó en los términos más frecuentes encontrados en PubTabNet y FinTabNet, combinados con texto aleatorio (Nassar et al., 2022). Los dos primeros conjuntos de datos sintéticos fueron afinados para replicar la apariencia de los conjuntos de datos originales, pero incorporando estructuras de tablas más complejas. El tercer conjunto adopta una apariencia colorida con alto contraste, mientras que el último contiene

tablas con contenido disperso. En última instancia, se han fusionado todos los conjuntos de datos sintéticos en un conjunto sintético unificado y público[1] que abarca un total de 600,000 ejemplos (Nassar et al., 2022).

- **Preparación de datos:** La fase de preparación de datos abarca todas las actividades para construir el conjunto de datos final (datos que se alimentarán en la(s) herramienta(s) de modelado) a partir de los datos iniciales en bruto. Las tareas de preparación de datos probablemente se realicen varias veces y no sigan un orden prescrito (Wirth & Hipp, 2000).

  Dado que *SynthTabNet* (Nassar et al., 2022) presenta un conjunto de imágenes de tablas delimitadas y anotadas, la preparación de los datos necesaria para el desarrollo de este proyecto únicamente necesita del filtrado del subconjunto de datos por utilizar, y de la limpieza de errores en la representación de anotaciones en los mismos.

- **Modelado:** En esta fase, se seleccionan y aplican diversas técnicas de modelado y se calibran sus parámetros a valores óptimos. Por lo general, existen varias técnicas para el mismo tipo de problema de minería de datos. Algunas técnicas requieren formatos de datos específicos (Wirth & Hipp, 2000).

  Esta etapa tiene como objetivo seleccionar propuestas del estado del arte para su evaluación y realizar las transformaciones necesarias en sus inferencias para compararlas con los datos anotados previamente seleccionados.

- **Evaluación:** Antes de avanzar hacia la implementación final del modelo, es importante evaluar a fondo los resultados y revisar los pasos ejecutados para construirlo, asegurándose de que logre adecuadamente los objetivos comerciales. Un objetivo clave es determinar si hay algún problema comercial importante que no se haya considerado lo suficiente (Wirth & Hipp, 2000).

  Esta etapa necesita de un medio de comparación de resultados robusto, por lo que se propone identificar métricas de evaluación del estado del arte que reflejen con precisión el rendimiento de modelos para la detección de estructuras de tablas, e implementar un entorno de desarrollo de experimentos para comparar el rendimiento de los diferentes modelos seleccionados.

  La etapa de evaluación de este proyecto busca además contextualizar los resultados encontrados con las características de cada modelo, describiendo sus fortalezas y debilidades.

[1] https://github.com/IBM/SynthTabNet

- **Despliegue:** La creación del modelo generalmente no marca el fin del proyecto. Por lo general, es necesario organizar y presentar el conocimiento adquirido de una manera que el cliente pueda utilizar. La fase de implementación puede ser tan simple como generar un informe o tan compleja como implementar un proceso de minería de datos repetible, dependiendo de los requisitos. En cualquier caso, es crucial comprender de antemano las acciones necesarias para realmente aprovechar los modelos creados (Wirth & Hipp, 2000).

  La etapa final de este proyecto busca reunir los hallazgos encontrados. Es fundamentalmente un proceso de documentación que presenta el desempeño de los modelos bajo la(s) métrica(s) seleccionada(s), las fortalezas y debilidades de cada enfoque y las diversas conclusiones alcanzadas.

6 Modelos por comparar

Para el desarrollo de este trabajo, se seleccionaron dos de las principales propuestas del estado del arte actual para su comparación. Los modelos seleccionados cumplen con los siguientes criterios: reportan un nivel de precisión superior al 90% para la métrica elegida, son de código abierto y publican los pesos de los modelos preentrenados.

Table Transformer (*TATR*), propuesto por (Smock et al., 2021, 2023), cuenta con un repositorio claro y detallado (Smock & Pesala, 2021) donde publican el código para las etapas de entrenamiento e inferencia, y los pesos de los modelos preentrenados para ser utilizados a conveniencia. Tienen una amplia comunidad en *Hugging Face* y reportan buenos resultados con una métrica propietaria (Smock et al., 2022).

Unitable, propuesto por Peng et al. (2024), es una publicación reciente que presenta un paradigma de modelación diferente al de *Table Transformer*. *Unitable* ofrece un repositorio enfocado en facilitar su uso por otros equipos de investigación, publicando su código de entrenamiento e inferencia, un cuaderno de ejemplo para el uso del modelo en sus diferentes versiones y los pesos de los modelos preentrenados.[2].

6.1 Table Transformer

Smock et al. (2021) proponen la modelación de la detección de la tabla, el reconocimiento de su estructura y su análisis funcional como una tarea de detección de objetos con imágenes como entrada. Proponen modelar el reconocimiento de la estructura de tablas y su análisis funcional unificadamente con el uso de seis clases: tabla, columna (*column*), fila (*row*), encabezado de columna (*column header*), encabezado de fila proyectado (*projected row header*) y celdas extendidas (*spanning cell*), como se muestra en la Figura 4. La intersección de cada par de objetos de columna y fila forman implícitamente una nueva clase: la celda de cuadrícula de tabla. Ahora bien, la superposición física de estos objetos es la que finalmente representa la estructura jerárquica de una tabla (Smock et al., 2021).

[2] https://github.com/poloclub/unitable

Es importante mencionar que este modelo utiliza coordenadas dilatadas. Es decir, para cada par de filas adyacentes y cada par de columnas adyacentes, se expanden sus límites hasta que se encuentren a mitad de camino, llenando así el espacio vacío entre ellas. De manera similar, se expanden los objetos de las otras clases para que sus límites coincidan con los ajustes realizados a las filas y columnas que ocupan. Esto permite eliminar los espacios y solapamientos entre filas, columnas y celdas (Smock et al., 2021).

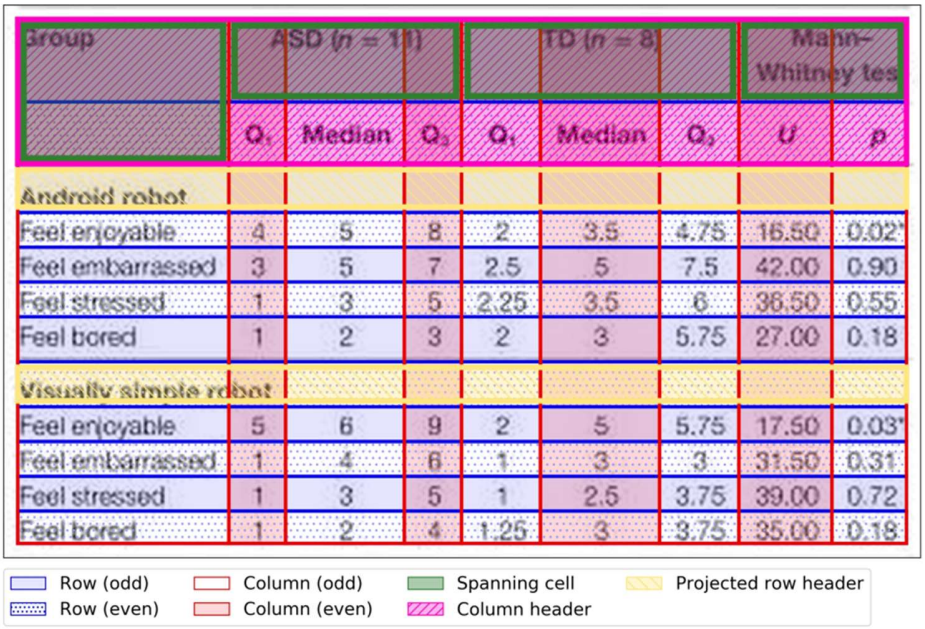

Figura 4. Ejemplo de anotaciones de coordenadas dilatadas para diferentes clases de objetos que modelan conjuntamente el reconocimiento de la estructura de tablas y su análisis funcional (Smock et al., 2021).

En cuanto a su arquitectura, este modelo entrena un *transformer* de detección (Carion et al., 2020) que utiliza una estructura ResNet-18 preentrenada en *ImageNet* con las primeras capas congeladas (Smock et al., 2021).

Es importante mencionar que Smock et al. (2023) desarrollaron una metodología para estandarizar diferentes conjuntos de datos de entrenamiento disponibles en la literatura de forma que puedan ser utilizados para entrenar *Table Transformer*. El modelo preentrenado que combina *PubTables-1M* y *FinTabNet* demostró ser superior al propuesto en 2021 y es entonces el que se usa para el desarrollo de este trabajo.

6.2 Unitable

Peng et al. (2024) define el objetivo del reconocimiento de tablas es traducir la imagen de una tabla de entrada I en una secuencia legible por una máquina T, que típicamente consiste en la estructura de la tabla S, el rectángulo de coordenadas de las celdas B y el contenido de las celdas C. La estructura $S = [s_1, \ldots, s_m]$ es una secuencia s de etiquetas HTML *tokenizadas*, el rectángulo de coordenadas de las celdas $B = [b_1, \ldots, b_n]$ es una secuencia de rectángulos de

coordenadas definidos por $b = (x_{min}, y_{min}, x_{max}, y_{max})$ y el contenido de las celdas $C = [c_1, \ldots, c_n]$ comprende el contenido dentro de cada celda en orden de lectura. Nótese que la longitud de la secuencia de B y C es la misma, pero más corta que la de S, ya que las etiquetas HTML contienen tanto celdas vacías como no vacías. Dado que cada celda i está definida por un único cuadro delimitador, c_i puede tener una o múltiples líneas de texto (Peng et al., 2024).

Unitable se compone de dos módulos principales: el codificador visual que procesa I y el decodificador de tareas que predice T (Peng et al., 2024).

El codificador visual extrae características de la imagen de entrada utilizando una proyección lineal que divide la imagen en parches (Peng et al., 2024). Si bien se ha demostrado que el uso de este tipo de proyecciones lineales limita la capacidad de atención cruzada del modelo sobre diferentes parches, empeorando su desempeño para el reconocimiento de tablas en comparación con modelos basados en estructuras convolucionales (Peng et al., 2023); *Unitable* propone compensar ese impacto con el uso de técnicas de preentrenamiento auto supervisadas (Peng et al., 2024).

En el proceso de preentrenamiento auto supervisado, se entrena el codificador visual para rellenar los parches enmascarados de imágenes tabulares prediciendo los tokens visuales correctos de un libro de códigos VQ-VAE (*Vector Quantized-Variational AutoEncoder*). Inicialmente, cada imagen tabular se divide en parches y una parte de estos parches se enmascara. La tarea del codificador es seleccionar los tokens apropiados del libro de códigos para reemplazar estas regiones enmascaradas. El libro de códigos VQ-VAE contiene un gran conjunto de tokens discretos, cada uno representando diferentes elementos visuales de la tabla, como espacios de fondo, texto y líneas de separación. El codificador aprende a reconstruir con precisión la imagen original al comprender y seleccionar los tokens correctos para cada parche. Este método aprovecha las representaciones de alto nivel de las estructuras y matices de las tablas capturadas por el VQ-VAE, entrenado en millones de imágenes tabulares, para asegurar una reconstrucción precisa y detallada de las áreas enmascaradas (Peng et al., 2024).

Así pues, el codificador visual se preentrena de manera autosupervisada y luego se le hace un refinamiento (*fine tunning*) junto con el decodificador de tareas en conjuntos de datos supervisados (Peng et al., 2024). El decodificador de tareas toma las características extraídas por el codificador visual y predice la secuencia de salida correspondiente (Peng et al., 2024).

Peng et al. (2024) entrenaron dos variantes de modelo: un modelo base con 30 millones de parámetros, incluyendo 4 capas de codificación, 8 cabezales de atención con un tamaño oculto de 512, y un modelo grande con 125M de parámetros, incluyendo 12 capas de codificación, 12 cabezales de atención con un tamaño oculto de 768. Tanto el modelo base como el grande tienen un decodificador de tareas con 4 capas de decodificación. La longitud máxima de la secuencia de tokens es de 512 para la estructura de la tabla, 1024 para el cuadro delimitador de

celdas y 200 para el contenido de las celdas, dado que encontraron que tales configuraciones satisfacen la mayoría de las tablas (Peng et al., 2024).

Para el desarrollo de este trabajo se utilizó el modelo grande (*Unitable-large*) puesto que Peng et al. (2024) muestran que presenta un mejor desempeño que el modelo base.

7 Experimentos

7.1 Descripción de los datos

SynthTabNet (Nassar et al., 2022) presenta un conjunto de tablas cuyas anotaciones continúan con la representación propuesta para los conjuntos de datos en los que se basa: *PubTabNet* (Zhong et al., 2019) y *FinTabNet* (Zheng et al., 2020). Las tablas se presentan como una estructura de árbol en el formato HTML. La raíz tiene dos hijos, "*thead*" y "*tbody*", que agrupan respectivamente los encabezados de la tabla y las celdas del cuerpo de la tabla. Los hijos de los nodos "*thead*" y "*tbody*" son las filas de la tabla (*tr*). Las hojas del árbol son las celdas de la tabla (*td*). Cada nodo de celda puede tener tres atributos: *"colspan"*, que indica el ancho de su columna*; "rowspan",* que indican el alto de su fila; y su contenido (Zhong et al., 2019). El formato HTML es útil dado que puede ser fácilmente utilizado en exploradores web y su representación en estructura de árbol permite utilizar la similitud basada en la distancia de edición de árboles (*TEDS*, del inglés *Tree-Edit-Distance-based Similarity*) como métrica de evaluación (Zhong et al., 2019).

Los datos anotados almacenan la estructura de la tabla de forma *tokenizada* como se evidencia en la Figura 5.

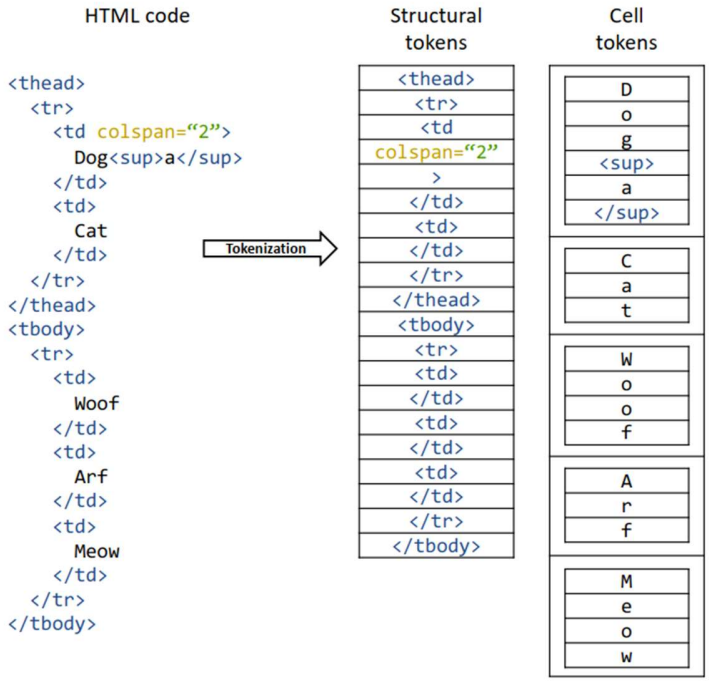

Figura 5. Ejemplo de representación *tokenizada* de una tabla (Zhong et al., 2019)

En cuanto a la organización de archivos, *SynthTabNet* separa las imágenes de referencia en cuatro categorías. *PubTabNet* y *FinTabNet* buscan replicar la apariencia de las tablas presentes en esos conjuntos de datos, pero incorporan estructuras más complejas. La categoría de *Marketing* presenta tablas en alto contraste y coloridas, y la categoría *Sparse* contiene tablas con contenido disperso (Nassar et al., 2022).

Como es usual para el trabajo con modelos de aprendizaje automático, *SynthTabNet* divide aleatoriamente cada categoría de los datos en los conjuntos de entrenamiento (*train*), prueba (*test*) y validación (*val*). Buscando reducir el trabajo computacional para el desarrollo de este proyecto, se utiliza como datos de referencia únicamente los correspondientes a la división de prueba (*test*).

Las anotaciones para cada categoría se presentan en un archivo "*synthetic_data.jsonl*" en donde cada línea corresponde a un documento JSON que reúne la información de cada anotación como se muestra en la Figura 6.

```
"filename": "png image filename inside one of the 'test', 'train', 'val'
directories",
"split": "One of 'test', 'train', 'val'",
"html": "Table structure and content",
    "cells": "Array with all table cells",
        "cell_id": "Zero based cell counter",
        "is_header": "true if that cell is part of the table header",
        "span": "In case there is a rowspan / columnspan",
            "spantype": "One of 'rowspan', 'colspan', '2dspan'. The '2dspan'
is used in case there is a rowspan and colspan in the same cell",
            "rowspan": "Number of rowspans for this cell",
            "colspan": "Number of colspans for this cell"
        "tokens": "Array with the tokenized content of the cell",
        "bbox": "The bounding box and the class of the cell in [x1, y1, x2,
y2, class] format"
    "structure":
        "tokens": "Array with html tags that describe the table structure"
```

Figura 6. Detalle de las anotaciones de *SynthTabNet* (Nassar et al., 2022)

El filtrado de los datos y la limpieza de la estructura de los JSONs anotados se realizó utilizando herramientas básicas de procesamiento de texto de Linux como *grep* y *awk* como se muestra en la Figura 7.

```
# Paso 1: Usar grep para filtrar líneas que contienen '"split": "test"' de
synthetic_data.jsonl.
grep '"split": "test"' synthetic_data.jsonl | \

# Paso 2: Usar awk para eliminar comas seguidas de llaves o corchetes de
cierre.
awk '{ print gensub(/,\s*([}\]])/, "\\1", "g") }' > test_synthetic_data.jsonl
```

Figura 7. Comando utilizado para el filtrado y limpieza de los datos anotados

7.2 Similitud basada en la Distancia de Edición de Árboles (*TEDS*)

La medida estándar para la similitud de árboles es la distancia de edición de árboles (*TED*, del inglés *Tree-Edit-Distance*), que se define como la secuencia de operaciones de edición de nodos de costo mínimo que transforma un árbol en otro (Pawlik & Augsten, 2016). Zhong et al. (2019) define entonces las siguientes operaciones:

- El costo de las operaciones de inserción y eliminación es 1.
- Cuando la edición es sustituir un nodo n_0 por n_s, el costo es 1 si n_0 o n_s no son celdas de la tabla (*td)*.
- Cuando tanto n_0 como n_s son *td*, el costo de sustitución es 1 si el ancho de columna o el alto de fila de n_0 y n_s es diferente. De lo contrario, el costo de sustitución es la similitud de Levenshtein (Levenshtein, 1965) entre el contenido de n_0 y n_s normalizada entre 0 y 1.

Finalmente, la ecuación (1) muestra el cálculo de la $TEDS$ entre dos árboles.

$$\mathrm{TEDS}(\mathrm{T_a}, \mathrm{T_b}) = 1 - \frac{\mathrm{EditDist}(\mathrm{T_a}, \mathrm{T_b})}{\max(|\mathrm{T_a}|, |\mathrm{T_b}|)} \quad (1)$$

Donde $EditDist$ denota la distancia de edición entre los árboles y $|T_i|$ es el número de nodos en el árbol T_i. El rendimiento del reconocimiento de tablas de un método a evaluar en un conjunto de muestras de prueba se define como la $TEDS$ promedio entre el resultado del reconocimiento y los datos reales de cada muestra (Zhong et al., 2019).

Como este trabajo se basa únicamente en el reconocimiento de la estructura de tablas, se utilizará $S - TEDS$, una modificación de $TEDS$ propuesta por Huang et al. (2023) que solo considera la estructura lógica de una tabla e ignora el contenido de sus celdas.

7.3 Comparación de modelos

Buscando estandarizar la comparación de modelos, se propone implementar una capa de abstracción sobre cada proyecto que permita realizar inferencias sobre todos los datos de prueba con cada modelo, calcular su $S - TEDS$ asociada y almacenar los resultados de forma ordenada en un archivo CSV.

Esta capa de abstracción agrupa la interacción con los modelos en la clase *TestModel* como se presenta en el Algoritmo 1. Esta clase define durante su inicialización la preparación de las imágenes de referencia y la reconstrucción de la estructura tokenizada de las tablas en código HTML, como se muestra en el

Algoritmo 2. *TestModel* también agrupa las funciones para ejecutar la inferencia del modelo a probar y el cálculo de la $S-TEDS$ para cada imagen utilizada.

La clase *TestModel* interactúa con el modelo a evaluar por intermedio de una clase ejecutora en donde se reúnen los componentes fundamentales para realizar inferencias en modo de detección de estructura de tablas. Esta clase cuenta con dos componentes principales: la inicialización, en donde se carga la estructura del modelo a utilizar y se configura el hardware; y la predicción, que reúne el conjunto de métodos y transformaciones necesarios para adaptar la imagen a procesar de acuerdo con los requisitos del modelo, realizar la inferencia y transformar la salida del modelo en formato HTML. Esto con el fin de facilitar la medición de su desempeño utilizando la $S-TEDS$.

Finalmente, se recorre iterativamente cada tabla de referencia, se calcula su métrica y los resultados son almacenados en un archivo CSV para su posterior análisis. Este proceso se describe en el Algoritmo 3.

Dado que cada modelo tiene requerimientos, entornos y modos de operación diferentes, se ejecutaron en proyectos independientes donde se implementó una capa de abstracción uniforme sobre el código base ofrecido por sus autores. En cada proyecto, se define una clase ejecutora para una versión simplificada del modelo a evaluar, con el fin de realizar inferencias únicamente sobre la estructura de las tablas. Posteriormente, se ejecuta la evaluación de los datos y se almacenan los resultados.

Algoritmo 1. Estructura de la clase *TestModel*.

```
1:  CLASS TestModel:
2:    METHOD __init__(self, image_path, ground_truth_data, padding=0):
3:      Load image from image_path and convert to RGB
4:      if padding > 0:
5:        Expand image with white border
6:      anno_code ← process ground_truth_data(ground_truth_data)
7:    METHOD inference(self, model):
8:      Generate prediction code from model using image
9:      Format prediction code as HTML
10:     return formatted prediction code
11:   METHOD compute_metric(self):
12:     Initialize metric for structure evaluation
13:     Evaluate metric using prediction and annotation codes
14:     return metric value
```

Algoritmo 2. Preparación de datos de referencia.

```
FUNCTION process_ground_truth_data(ground_truth_data)
    Extract html structure tokens and cell tokens from ground_truth_data
    Initialize anno_html and idx
    while idx < length of anno_html_raw do
        if current token indicates a cell start then
            Append combined cell start and end tag to anno_html
            Increment idx by 2
        else
            Append current token to anno_html
            Increment idx by 1
        end if
    end while
    Clean and process cell tokens into anno_cell
    Combine processed structure and cell tokens into anno_code
    return anno_code
```

Algoritmo 3. Función de evaluación de los datos de referencia.

```
FUNCTION evaluate_ground_truth()
    Get list of directories in DATA_PATH
    for each directory in directories do
        Initialize results list
        Construct ground_truth_file path
        Open ground_truth_file and read line by line
        Initialize count
        for each line in file do
            Construct image_path from directory and ground_truth_data
            Initialize TestModel with image_path and ground_truth_data
            Generate prediction using TestModel
            Store elapsed time
            Compute metric using TestModel
            Append results to results list
        end for
        Log total files processed for directory
        Save results to CSV file
    end for
```

7.4 Ejecución de modelos

Con respecto a *Table Transformer*, Smock et al. (2023) presentaron en su repositorio (Smock & Pesala, 2021) scripts con ejemplos de inferencia y enlaces para acceder a los modelos preentrenados. Es importante mencionar que se utilizó el último modelo publicado en agosto 2023, que utiliza una ampliación en los datos de entrenamiento con respecto a versiones anteriores del mismo proyecto. Esta información se tomó como punto de partida para establecer la clase ejecutora y la capa de evaluación presentada en la sección anterior.

La transformación a HTML presentada por Smock et al. (2023) no correspondía con la estructura utilizada en *SynthTabNet*, por lo que fue necesario implementar una función que adaptara la salida del modelo a un código HTML comparable dentro de la clase ejecutora para este modelo.

Con respecto a *Unitable*, Peng et al. (2024) presenta una guía detallada de puesta en marcha de *Unitable* en su repositorio[3]. Esta guía fue utilizada como punto de referencia para establecer la clase ejecutora de este modelo y su capa de evaluación correspondiente. El código presentado por Peng et al. (2024) también fue utilizado para la implementación de las funciones de cálculo de la *TEDS*. Si bien su implementación permite comparar el contenido de las tablas, para el desarrollo de este trabajo únicamente se utilizó la $S-TEDS$.

La implementación final para la evaluación de este modelo puede ser consultada en el repositorio de este proyecto (Garzón Vargas, 2024)[4].

7.5 Resultados obtenidos

Se calculó el valor promedio de la $S-TEDS$ para cada categoría de *SynthTabNet* por modelo. En el repositorio del proyecto están publicados los archivos CSV que detallan el desempeño de cada modelo para cada categoría de los datos. Estos fueron resumidos en la Tabla 2, en donde también se calcula un promedio general del valor de la métrica como parámetro que resume el desempeño del modelo sobre el conjunto completo de datos utilizado para la detección de estructura de tablas.

A manera de ejemplo, desde la Figura 8 a la Figura 16 se muestran imágenes del conjunto de datos de prueba y la renderización del código HTML inferido por cada uno de los modelos a comparar para cada una de ellas. Se evidencia como *Unitable* percibe de mejor forma detalles complejos como variación en el ancho y alto de una celda, encabezados de múltiples líneas y dimensiones variadas y la representación de tablas dispersas.

La Tabla 2 muestra el cálculo de la $S-TEDS$ por categoría para cada uno de los modelos evaluados. Se evidencia la superioridad de *Unitable* (Peng et al., 2024) sobre *Table Transformer* (Smock et al., 2023) para el reconocimiento de estructuras tabulares en imágenes de tablas.

Tabla 2. *S-TEDS* promedio para los datos analizados.

	S-TEDS				
***SynthTabNet* **	***FinTabNet***	***Marketing***	***PubTabNet***	***Sparse***	**Promedio**
Table Transformer	0.8288	0.7047	0.8341	0.6166	0.7460
Unitable	0.9959	0.9911	0.9957	0.9938	0.9941

[3] https://github.com/poloclub/unitable
[4] https://github.com/JmGarzon/TSR_model_comparison

Some amounts may not reconcile due to rounding.	Operating expenses	Tax credits		December 31, 2008	Construction in progress	Dividends paid
Other accrued liabilities	Interest Rate	Revenue:		Total	Executive Compensation	Gross Unrealized Losses
Depreciation, depletion and amortization	December 31, 2007	Net cash provided by used in investing activities	Southwest	Cash provided by operating activities	Money market funds	Current:
Pension Plans	First	Baa3		State and local	Exhibit	In thousands
Land	Preferred stock	Current:	Balance, end of period	Interest rate	Prepaid expenses and other current assets	Second quarter

Figura 8. Ejemplo de tabla anotada: *FinTabNet*.

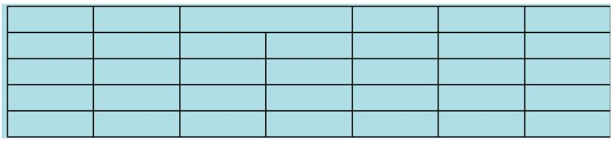

Figura 9. Representación HTML de la predicción de *Unitable* para un ejemplo de " *FinTabNet* ". *S-TEDS*: 1.0

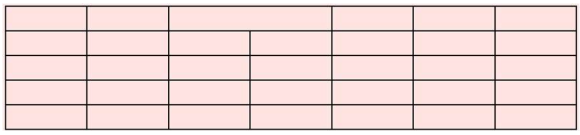

Figura 10. Representación HTML de la predicción de *TATR* para un ejemplo de "*FinTabNet*". *S-TEDS*: 1.0

Mean SD	B	P-value	85.82%	Group
n Strain			Year	
No	.	no	M	nd
N		Mean	60930.51$	III
Percentage	✓	Low	%	yes

Figura 11. Ejemplo de tabla anotada: *PubTabNet*.

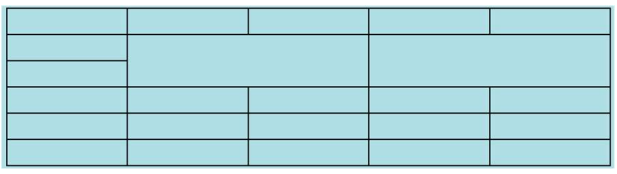

Figura 12. Representación HTML de la predicción de *Unitable* para un ejemplo de " *PubTabNet* ". *S-TEDS*: 0.89.

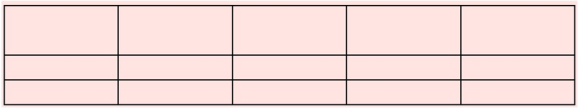

Figura 13. Representación HTML de la predicción de *TATR* para un ejemplo de " *PubTabNet* ". *S-TEDS*: 0.6.

Patient Type Region	Frequency Method	Group Cases	95 % CI	Gene Region	Age Group All	Age Patient	Age Odds ratio df
Description	42						
Model		89					
Number						65	
Sex							
Mean							
Treatment	81		3		45		95
Sample				75			18
p-value				91			
Sex			98				
Mean SD							5
							15
Name							
Baseline	34					9	
Cases							
Mean	29						73
P value		91			20		78

Figura 14. Ejemplo de tabla anotada: *Sparse*.

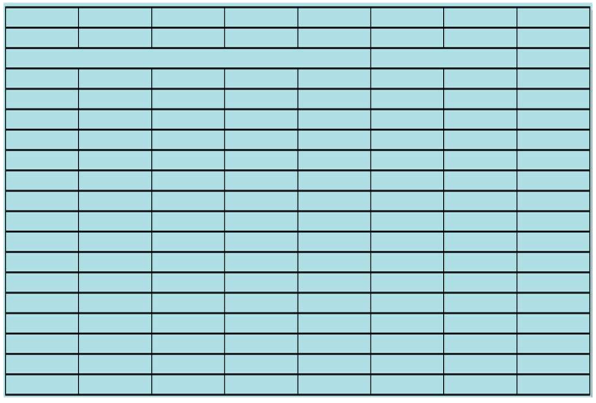

Figura 15. Representación HTML de la predicción de *Unitable* para un ejemplo de " *Sparse* ". *S-TEDS*: 0.99

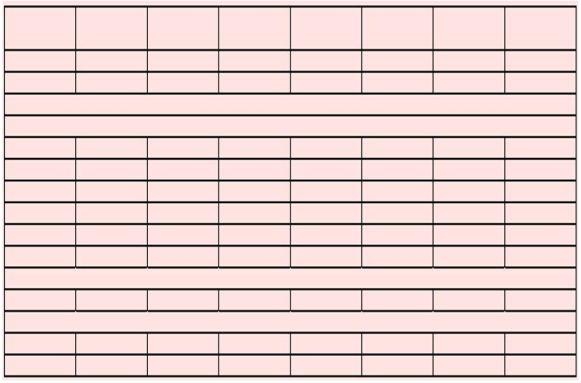

Figura 16. Representación HTML de la predicción de *TATR* para un ejemplo de "*Sparse* ". *S-TEDS*: 0.68.

Thereafter		Discount rate	324886	Service cost	Revenue		$-	Thereafter	In thousands	
	Revenue	State	Third Quarter		Interest income	-	Europe	Years Ended December 31,	Canada	
	68.8%	State	Fourth Quarter	—	Operating income		Long-term debt	United States	431.32	10-K
Other	Description	Net sales	$-	In millions	$-	—			Gross profit	
December 31,	Leased	*	Deferred income taxes	Total					International	Fair Value
Total revenues	Total	N/A		Interest expense			United States		$-	
X	Total	Interest cost	Income from continuing operations	% Change	Leased					
International	Service cost			85.37%					Third Quarter	
Forfeited	In thousands	—%	Interest cost				10-Q	Canada	Fair Value	
Long-term debt	Provision for income taxes	Other assets	-	In thousands	Gross profit			10-K	Years Ended December 31,	X
$—	Europe	Service cost	X	In thousands				NM	December 31,	
n/a	Amount	$-	N/A			n/a	Interest cost		Other, net	
Goodwill	Depreciation and amortization	—%	Other assets				in thousands	Second Quarter	In thousands	
Years Ended December 31,	Other assets	International	Total assets	Basic		Granted	Service cost	Interest cost	NA	Exercised
-					Cash and cash equivalents	Revenues				Diluted

Figura 17. Ejemplo de tabla anotada: *Marketing*.

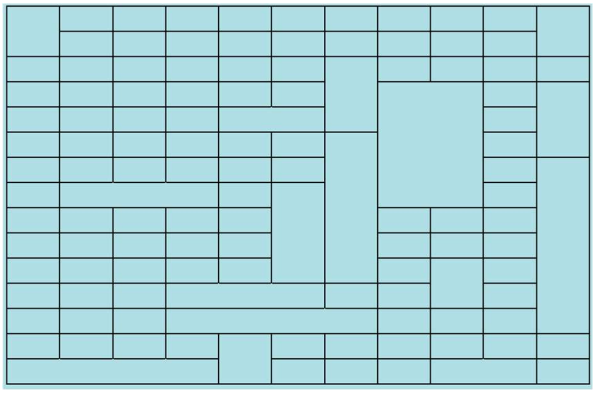

Figura 18. Representación HTML de la predicción de *Unitable* para un ejemplo de "*marketing*".
S-TEDS: 1.0

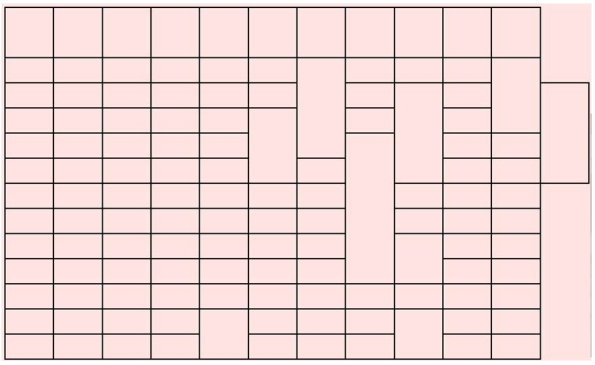

Figura 19. Representación HTML de la predicción de *TATR* para un ejemplo de "*marketing*".
S-TEDS: 0.677

8 Conclusiones

La comparación de modelos puede ser una tarea desafiante, ya que la estandarización de entradas y salidas para hacerlas comparables no es trivial y requiere un alto nivel de comprensión del funcionamiento de los modelos a evaluar. La diversidad en las arquitecturas y propósitos de los modelos añade complejidad, lo que hace indispensable un análisis minucioso de sus entradas y salidas para garantizar una evaluación justa y precisa.

La implementación de una capa de abstracción para el manejo homogéneo de modelos demostró ser una estrategia útil y efectiva en el desarrollo del proyecto. Esta capa permitió modularizar los diferentes componentes involucrados, facilitando la depuración y estandarizando los resultados para su posterior análisis. Este enfoque no solo optimiza el flujo de trabajo, sino que también mejora la reproducibilidad y la escalabilidad del proyecto. Se espera que esta herramienta pueda ser extendida para comparar diversas variaciones de los modelos base, y que sus hallazgos puedan ser utilizados para complementar sistemas generales de extracción de información de documentos digitalizados.

En cuanto a la evaluación de modelos específicos, se evidenció la superioridad de *Unitable* sobre *Table Transformer* en la representación de tablas complejas, que son el foco de estudio de *SynthTabNet*. Una posible explicación de esta diferencia es que *Table Transformer* intenta realizar un análisis funcional de las celdas en el mismo modelo que infiere la estructura. La combinación de estas tareas en un solo modelo podría aumentar la complejidad y dificultar la generalización de la inferencia, afectando su desempeño en contextos más complejos.

Finalmente, se destaca la importancia de publicar proyectos relacionados con la ciencia de los datos con una buena documentación en su código. Ambos repositorios consultados para el desarrollo de este trabajo contaban con documentación de alta calidad, lo cual fue fundamental para el progreso del proyecto. Una documentación clara y detallada no solo facilita la comprensión y uso de los recursos, sino que también fomenta la colaboración y el avance dentro de la comunidad científica.

9 Referencias

Bharati Puja and Pramanik, A. (2020). Deep Learning Techniques—R-CNN to Mask R-CNN: A Survey. En J. and N. B. and P. S. K. and P. D. Das Asit Kumar and Nayak (Ed.), *Computational Intelligence in Pattern Recognition* (pp. 657–668). Springer Singapore.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). *End-to-End Object Detection with Transformers*. http://arxiv.org/abs/2005.12872

Coüasnon, B., & Lemaitre, A. (2014). Recognition of tables and forms. En *Handbook of Document Image Processing and Recognition* (pp. 647–677). Springer London. https://doi.org/10.1007/978-0-85729-859-1_20

Embley, D. W., Hurst, M., Lopresti, D., & Nagy, G. (2006). Table-processing paradigms: A research survey. En *International Journal on Document Analysis and Recognition* (Vol. 8, Números 2–3, pp. 66–86). https://doi.org/10.1007/s10032-006-0017-x

Fernandes, J., Xiao, B., Simsek, M., Kantarci, B., Khan, S., & Alkheir, A. A. (2023). TableStrRec: framework for table structure recognition in data sheet images. *International Journal on Document Analysis and Recognition*. https://doi.org/10.1007/s10032-023-00453-8

Fischer, P., Smajic, A., Mehler, A., & Abrami, G. (2021). *Multi-Type-TD-TSR -- Extracting Tables from Document Images using a Multi-stage Pipeline for Table Detection and Table Structure Recognition: from OCR to Structured Table Representations*.

Gao, L., Huang, Y., Dejean, H., Meunier, J. L., Yan, Q., Fang, Y., Kleber, F., & Lang, E. (2019). ICDAR 2019 competition on table detection and recognition (cTDaR). *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, 1510–1515. https://doi.org/10.1109/ICDAR.2019.00243

Garzón Vargas, J. M. (2024). *TSR_model_comparison*. https://github.com/JmGarzon/TSR_model_comparison

Göbel, M., Hassan, T., Oro, E., & Orsi, G. (2012). A methodology for evaluating algorithms for table understanding in PDF documents. *DocEng 2012 - Proceedings of the 2012 ACM Symposium on Document Engineering*, 45–48. https://doi.org/10.1145/2361354.2361365

Gobel, M., Hassan, T., Oro, E., & Orsi, G. (2013). ICDAR 2013 table competition. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, 1449–1453. https://doi.org/10.1109/ICDAR.2013.292

Hashmi, K. A., Liwicki, M., Stricker, D., Afzal, M. A., Afzal, M. A., & Afzal, M. Z. (2021). Current Status and Performance Analysis of Table Recognition in Document Images with Deep Neural Networks. *IEEE Access*, *9*, 87663–87685. https://doi.org/10.1109/ACCESS.2021.3087865

Holeček, M., Hoskovec, A., Baudiš, P., & Klinger, P. (2019). *Table understanding in structured documents*. https://doi.org/10.1109/ICDARW.2019.40098

Huang, Y., Lu, N., Chen, D., Li, Y., Xie, Z., Zhu, S., Gao, L., & Peng, W. (2023). *Improving Table Structure Recognition with Visual-Alignment Sequential Coordinate Modeling*. http://arxiv.org/abs/2303.06949

Kashinath, T., Jain, T., Agrawal, Y., Anand, T., & Singh, S. (2022). End-to-end table structure recognition and extraction in heterogeneous documents. *Applied Soft Computing*, *123*. https://doi.org/10.1016/j.asoc.2022.108942

Lee, E., Park, J., Koo, H. Il, & Cho, N. I. (2022). Deep-learning and graph-based approach to table structure recognition. *Multimedia Tools and Applications*, *81*(4), 5827–5848. https://doi.org/10.1007/s11042-021-11819-7

Levenshtein, V. I. (1965). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics. Doklady*, *10*, 707–710. https://api.semanticscholar.org/CorpusID:60827152

Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2022). A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(12), 6999–7019. https://doi.org/10.1109/TNNLS.2021.3084827

Lin, W., Sun, Z., Ma, C., Li, M., Wang, J., Sun, L., & Huo, Q. (2022). TSRFormer: Table Structure Recognition with Transformers. *MM 2022 - Proceedings of the 30th ACM International Conference on Multimedia*, 6473–6482. https://doi.org/10.1145/3503161.3548038

Nassar, A., Livathinos, N., Lysak, M., & Staar, P. (2022). *TableFormer: Table Structure Understanding with Transformers*. http://arxiv.org/abs/2203.01017

Pawlik, M., & Augsten, N. (2016). Tree edit distance: Robust and memory-efficient. *Information Systems*, *56*, 157–173. https://doi.org/10.1016/j.is.2015.08.004

Peng, S., Chakravarthy, A., Lee, S., Wang, X., Balasubramaniyan, R., & Chau, D. H. (2024). *UniTable: Towards a Unified Framework for Table Recognition via Self-Supervised Pretraining*. http://arxiv.org/abs/2403.04822

Peng, S., Lee, S., Wang, X., Balasubramaniyan, R., & Chau, D. H. (2023). *High-Performance Transformers for Table Structure Recognition Need Early Convolutions*. http://arxiv.org/abs/2311.05565

Prasad, D., Gadpal, A., Kapadni, K., Visave, M., & Sultanpure, K. (2020). *CascadeTabNet: An approach for end to end table detection and structure recognition from image-based documents*.

Riba, P., Dutta, A., Goldmann, L., Fornes, A., Ramos, O., & Llados, J. (2019). Table detection in invoice documents by graph neural networks. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, 122–127. https://doi.org/10.1109/ICDAR.2019.00028

Schröer, C., Kruse, F., & Gómez, J. M. (2021). A systematic literature review on applying CRISP-DM process model. *Procedia Computer Science*, *181*, 526–534. https://doi.org/10.1016/j.procs.2021.01.199

Shelhamer, E., Long, J., & Darrell, T. (2016). *Fully Convolutional Networks for Semantic Segmentation*. http://arxiv.org/abs/1605.06211

Silva, A. C., Jorge, A. M., & Torgo, L. (2006). Design of an end-to-end method to extract information from tables. *International Journal on Document Analysis and Recognition*, *8*(2–3), 144–171. https://doi.org/10.1007/s10032-005-0001-x

Šimsa, Š., Šulc, M., Uřičář, M., Patel, Y., Hamdi, A., Kocián, M., Skalický, M., Matas, J., Doucet, A., Coustaty, M., & Karatzas, D. (2023). *DocILE Benchmark for Document Information Localization and Extraction*. http://arxiv.org/abs/2302.05658

Smock, B., & Pesala, R. (2021). *Table Transformer*. https://github.com/microsoft/table-transformer

Smock, B., Pesala, R., & Abraham, R. (2021). *PubTables-1M: Towards comprehensive table extraction from unstructured documents*. http://arxiv.org/abs/2110.00061

Smock, B., Pesala, R., & Abraham, R. (2022). *GriTS: Grid table similarity metric for table structure recognition*. http://arxiv.org/abs/2203.12555

Smock, B., Pesala, R., & Abraham, R. (2023). Aligning benchmark datasets for table structure recognition. *International Conference on Document Analysis and Recognition*, 371–386.

Staudemeyer, R. C., & Morris, E. R. (2019). *Understanding LSTM -- a tutorial into Long Short-Term Memory Recurrent Neural Networks*. http://arxiv.org/abs/1909.09586

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention Is All You Need*. http://arxiv.org/abs/1706.03762

Wirth, R., & Hipp, J. (2000). CRISP-DM: Towards a standard process model for data mining. *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining*.

Xue, W., Li, Q., & Tao, D. (2019). ReS2TIM: Reconstruct Syntactic Structures from Table Images. *2019 International Conference on Document Analysis and Recognition (ICDAR)*, 749–755. https://doi.org/10.1109/ICDAR.2019.00125

Xue, W., Yu, B., Wang, W., Tao, D., & Li, Q. (2021). *TGRNet: A Table Graph Reconstruction Network for Table Structure Recognition*.

Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. En *Insights into Imaging* (Vol. 9, Número 4, pp. 611–629). Springer Verlag. https://doi.org/10.1007/s13244-018-0639-9

Zheng, X., Burdick, D., Popa, L., Zhong, X., & Wang, N. X. R. (2020). *Global Table Extractor (GTE): A Framework for Joint Table Identification and Cell Structure Recognition Using Visual Context*. http://arxiv.org/abs/2005.00589

Zhong, X., ShafieiBavani, E., & Jimeno-Yepes, A. (2019). Image-based table recognition: data, model, and evaluation. *CoRR*, *abs/1911.10683*. http://arxiv.org/abs/1911.10683

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., & Sun, M. (2018). *Graph Neural Networks: A Review of Methods and Applications*. http://arxiv.org/abs/1812.08434