Extracción Estructurada de Documentos con Deep Learning

Dr. Cristian Muñoz Villalobos

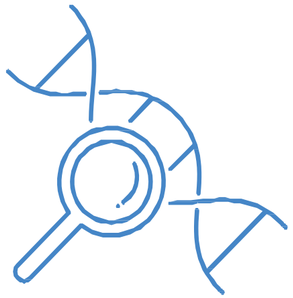
Introducción

La extracción de documentos estructurados es una tarea importante en muchas organizaciones y áreas de investigación. Encaja como un proceso de organización de datos (Data Curation) donde la recolección y gestión de datos son pasos clave para tener un fácil acceso que permita encontrar y comprender los datos.

Negocios

Normas

Economia

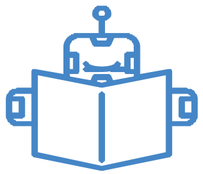


Biomedicina

De los enfoques para automatizar la organización de datos (Data Curation)

En términos generales, tenemos dos formas de automatizar el proceso de curación de datos:

- Podemos escribir un conjunto de instrucciones que le digan a la computadora cómo procesar la fecha, definiendo un conjunto de reglas.



- Podemos dejar que la computadora aprenda a procesar la fecha a partir de ejemplos empíricos, utilizando el aprendizaje profundo.

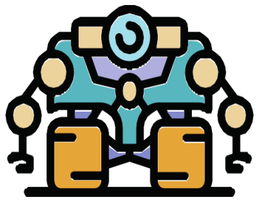
Enfoques basados en reglas

- Los enfoques basados en reglas tienen la ventaja de ser fáciles de entender. La mayoría de nosotros estamos acostumbrados a interactuar con la computadora a través de enfoques basados en reglas.

- Ciertamente tienen su importancia, pero hemos identificado que presentan poco rendimiento como por ejemplo: los documentos históricos.

- La complejidad y el ruido son enemigos de las reglas, y muchos tipos de documentos presentan muchas situaciones complejas y ruidosas.

Deep Learning



Aprende un mapeamiento robusto entre los datos brutos y la salida deseada.

Generaliza de forma que podamos procesar con precisión los nuevos datos a los cuales el modelo no fue expuesto durante el entrenamiento.

Los modelos raramente se entrenan desde cero.

Como la IA participa en la extracción estructurada de documentos?

Necesitamos detectar y extraer:

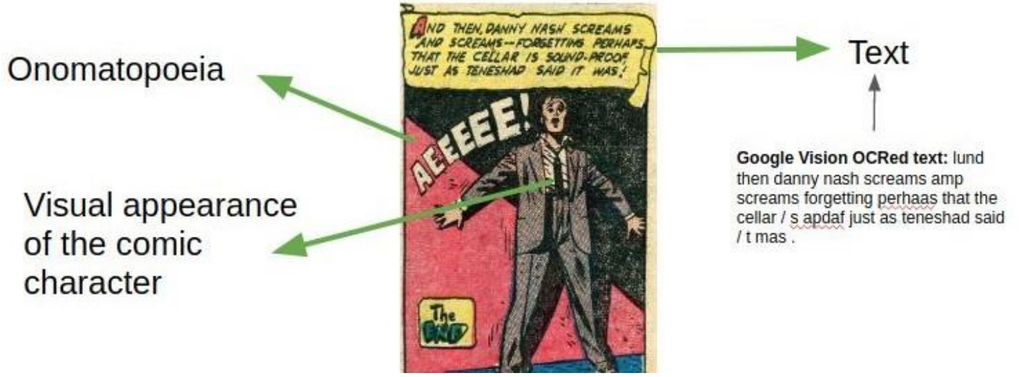
- Texto (Escaneado es del tipo más complicado)
  Título? subtítulo? Pie de página? encabezado?
- Tablas
- Imágenes
- Ecuaciones
- Manuscritos (Textos o firmas)
- Informaciones clave-valor

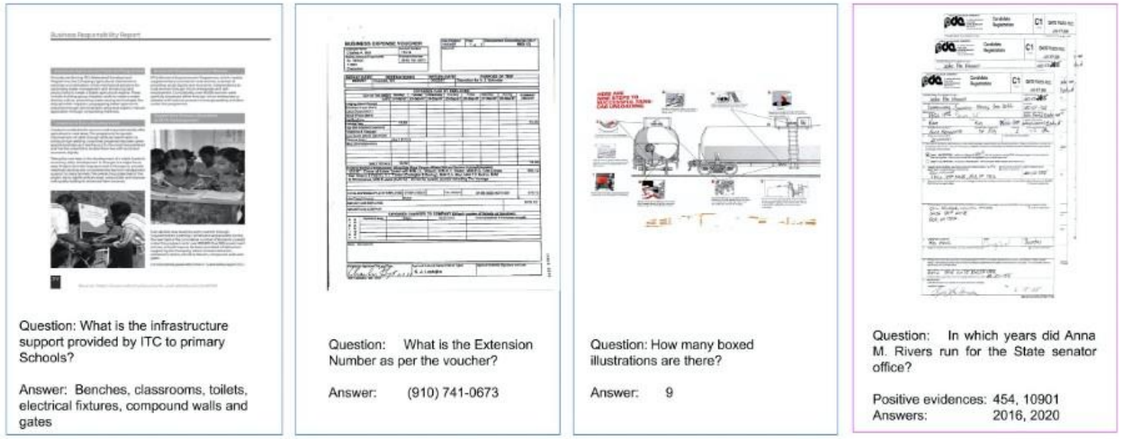
https://icdar2021.org/program-2/competitions/competition-on-scientific-literature-parsing/

- Análisis de documentos históricos

- Reconocimiento de emociones en es cenas de Comics

- Sistemas Visuales Pregunta Respuesta para documentos





Ejemplo: OCR Comerciales

- Los softwares comerciales de OCR se basan principalmente en documentos limpios y modernos con diseños simples, como libros de una columna.

- En la práctica, a menudo nos interesan los documentos ruidosos, muchos de los cuales tienen diseños muy complejos.

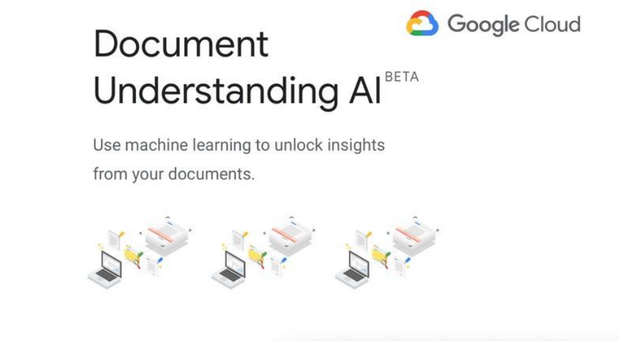
- Las soluciones de OCR existentes a menudo tienen problemas con los diseños y las fuentes.

- Estamos muy lejos de llegar a la Inteligencia Artificial General. Si el software no ha sido entrenado en documentos que se parecen a los documentos que desea procesar, tendrán un desempeño deficiente.

No hay un aplicativo que haga eso isso?

- Muchas personas pueden pensar "pero no existe un aplicativo(o algún otro producto comercial) que haga eso?
- Existen muchas empresas que ofrecen soluciones a la extracción de información:

https://cloud.google.com/document-ai



https://www.ibm.com/cloud/document-processing

https://aws.amazon.com/machine-learning/ml-use-cases/document-processing/

https://azure.microsoft.com/en-us/services/form-recognizer/#overview

- Desafortunadamente, no existen soluciones listas para usar para todos los tipos de conservación de datos (en CV o NLP), o las soluciones comerciales existentes a menudo no se acercan a una precisión aceptable.

- Una de las plataformas de código abierto más populares para resolver este tipo de problemas fue propuesta por la profesora Melissa Dell de la Universidad de Stanford.

Layout Parser

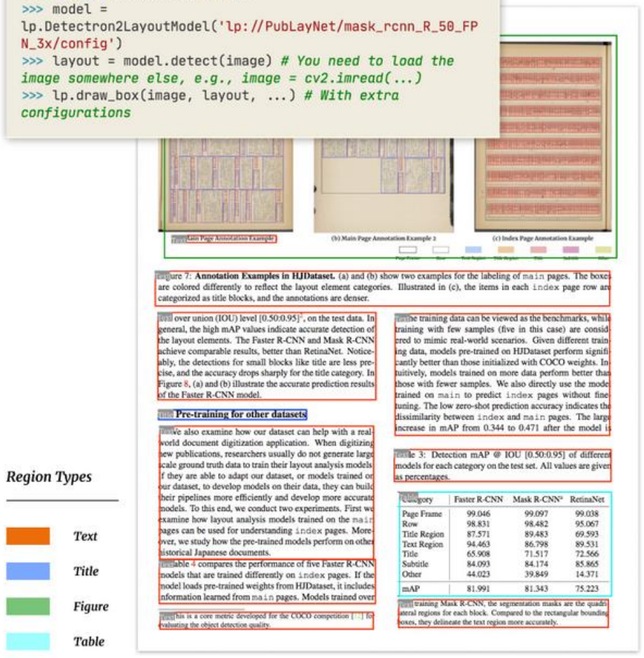
https://twitter.com/MelissaLDell/status/1380173059001307136

Layout Parser: Una solución Open Source

Layout Parser

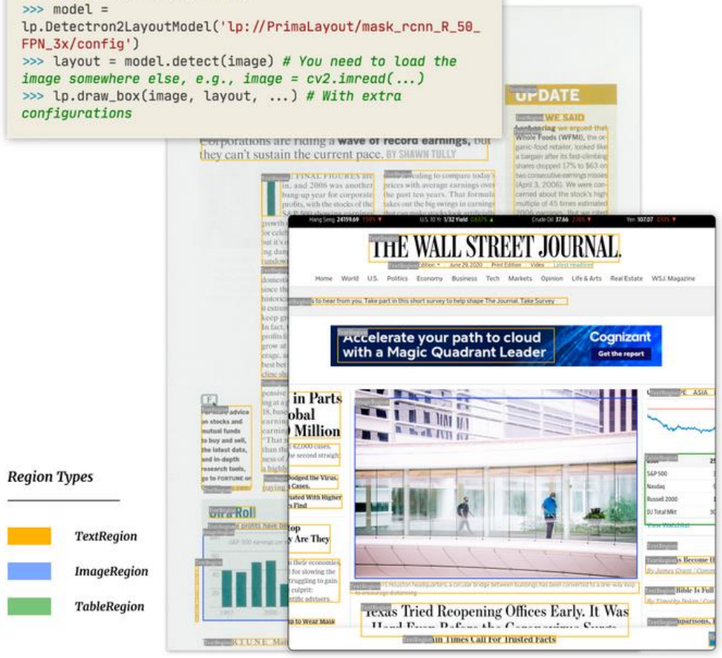
python

```
>>> import layoutparser as lp
>>> model =
lp.Detectron2LayoutModel('lp://PubLayNet/mask_rcnn_R_50_FPN_3x/config')
>>> layout = model.detect(image) # You need to load the image somewhere else, e.g., image = cv2.imread(...)
>>> lp.draw_box(image, layout, ...) # With extra configurations
```



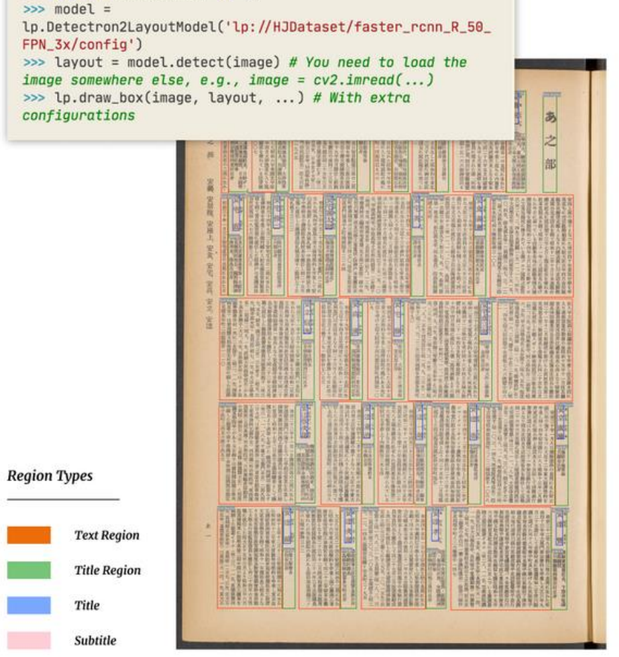
Paper with Complex Layouts

```
>>> import layoutparser as lp
>>> model =
lp.Detectron2LayoutModel('lp://PrimaLayout/mask_rcnn_R_50_FPN_3x/config')
>>> layout = model.detect(image) # You need to load the image somewhere else, e.g., image = cv2.imread(...)
>>> lp.draw_box(image, layout, ...) # With extra configurations
```



Magazine Scans & Websites

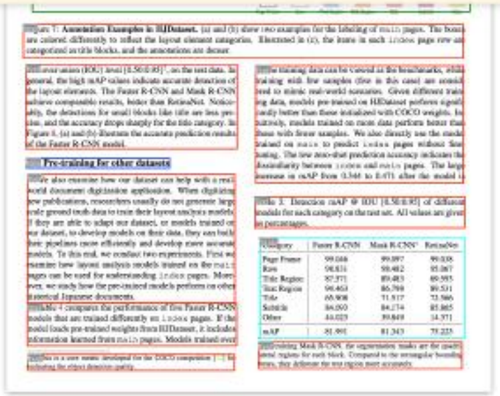
```
>>> import layoutparser as lp
>>> model =
lp.Detectron2LayoutModel('lp://HJDataset/faster_rcnn_R_50_FPN_3x/config')
>>> layout = model.detect(image) # You need to load the image somewhere else, e.g., image = cv2.imread(...)
>>> lp.draw_box(image, layout, ...) # With extra configurations
```



Historical Documents

Layout Parser

```
import layoutparser as lp
image = cv2.imread(...)
model = lp.Detector2LayoutModel('lp://PrimaLayout/mask_rcnn_R_50_FPN_3x/config')
layout = model.detect(image)
```



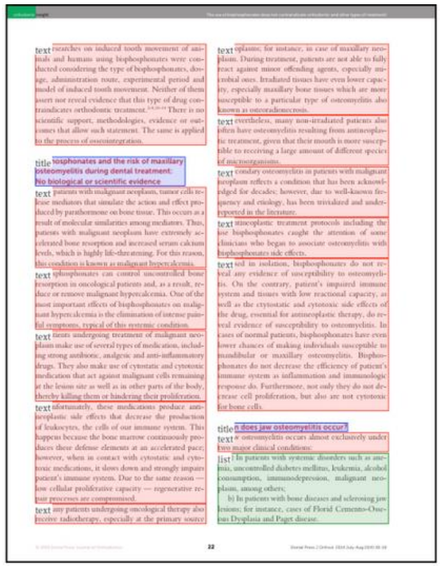
Paper with Complex Layouts

Magazine Scans & Websites

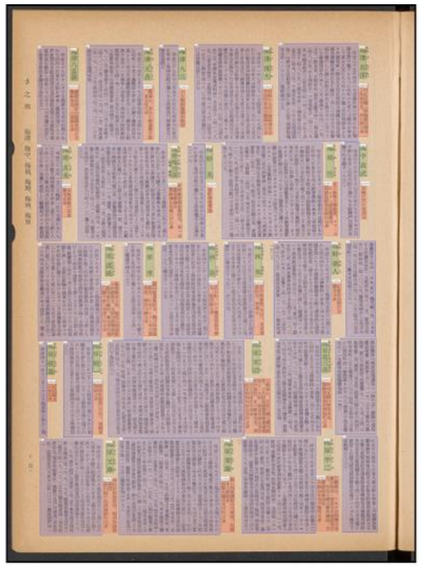
Historical Documents

Detección de composición de documentos

Una riqueza de modelos pre entrenado en conjuntos de datos diferentes

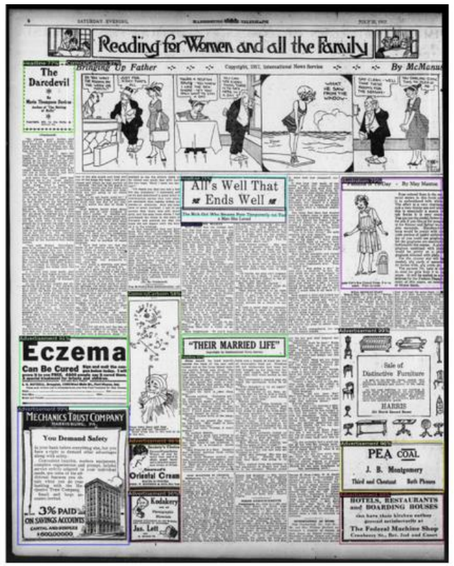


PubLayNet

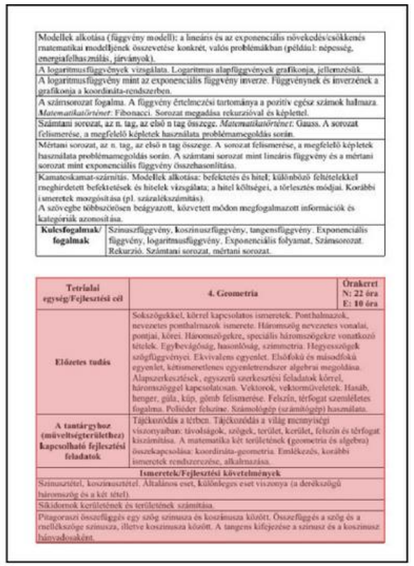


HJDataset

PRImA



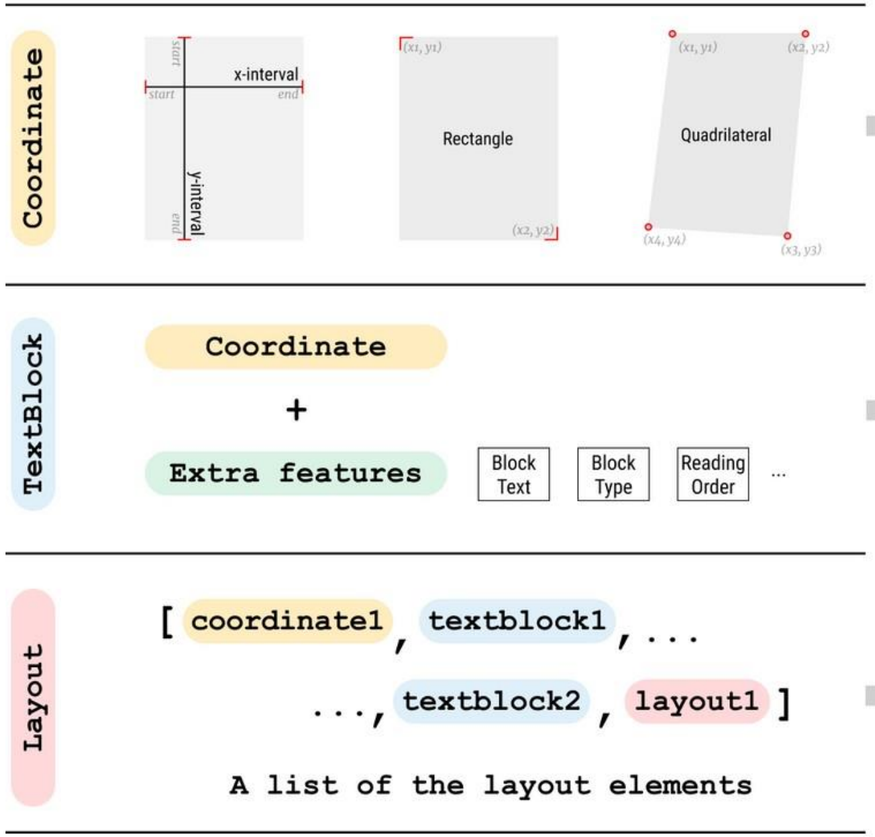
Newspaper Navigator



TableBank

Post-Processing

Manipulación y procesamiento de los componentes detectados:

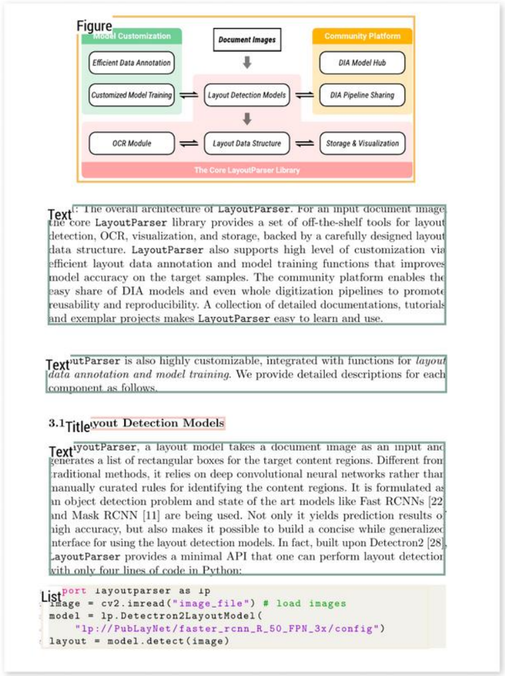


The same transformation and operation APIs

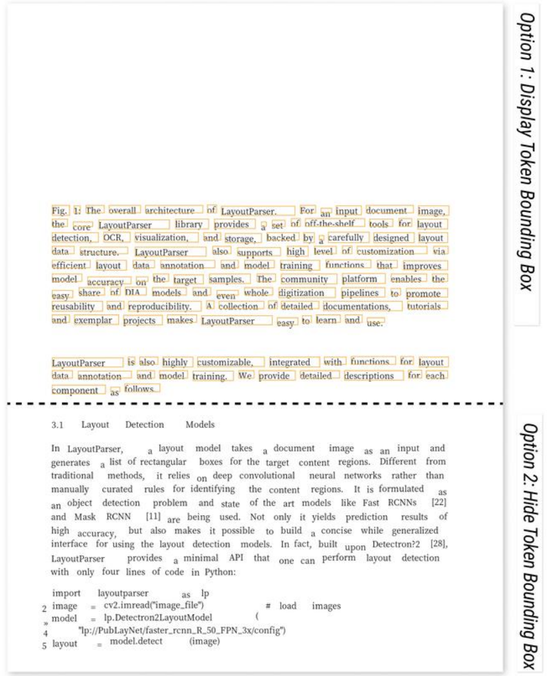
Operation Name	Description
`block.pad(top, bottom, right, left)`	Enlarge the current block according to the input
`block.scale(fx, fy)`	Scale the current block given the ratio in x and y direction
`block.shift(dx, dy)`	Move the current block with the shift distances in x and y direction
`block1.is_in(block2)`	Whether block1 is inside of block2
`block1.intersect(block2)`	Return the intersected region of block1 and block2. Coordinate type to be determined based on the inputs.
`block1.union(block2)`	Return the union region of block1 and block2. Coordinate type to be determined based on the inputs.
`block1.relative_to(block2)`	Convert the absolute coordinates of block1 to relative coordinates to block2
`block1.condition_on(block2)`	Calculate the absolute coordinates of block1 given the canvas block2's absolute coordinates
`block.crop_image(image)`	Obtain the image segments in the block region

Post-Processing

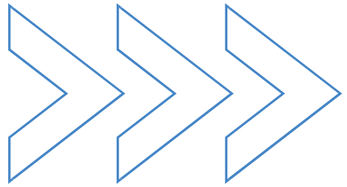
Visualizar, exportar y almacenar!

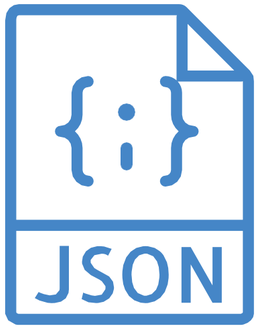


Mode I: Showing Layout on the Original Image



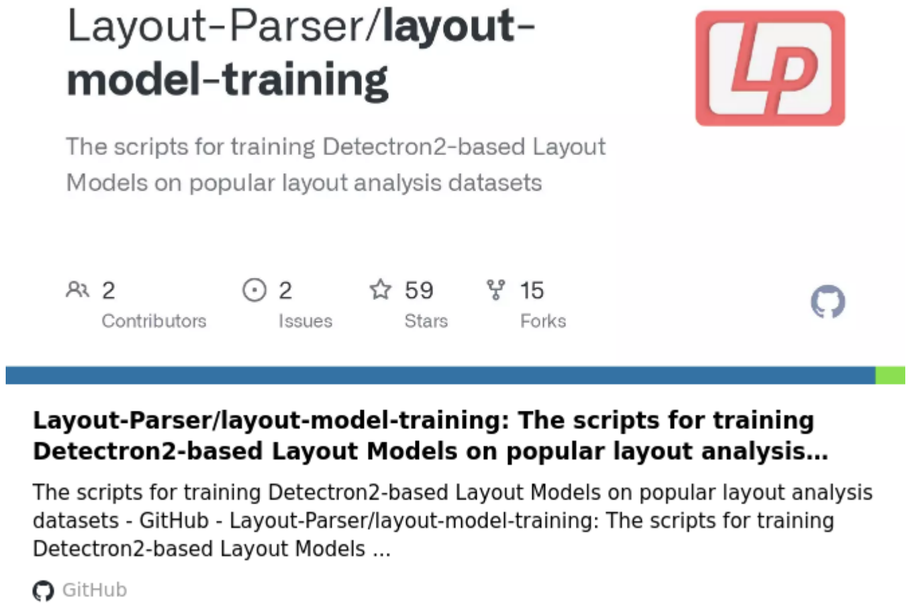
Mode II: Drawing OCR'd Text at the Correspoding Position





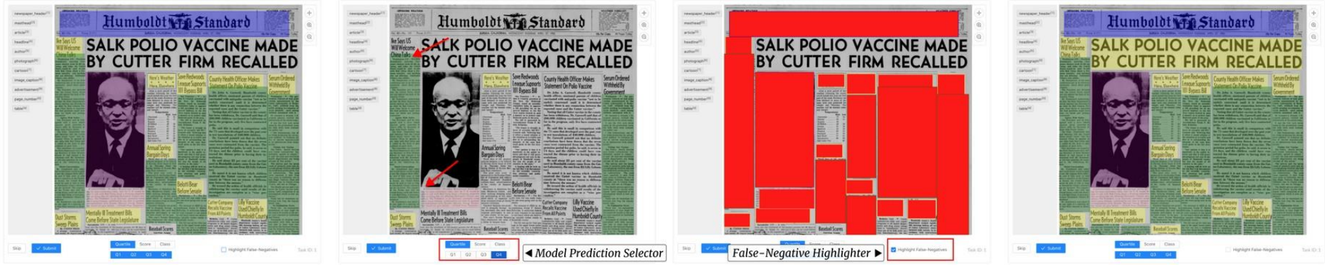
Personalización

- Entrenamiento de los modelos.



https://github.com/Layout-Parser/layout-model-training

- Anotación de datos eficientes



(a) Full Model Predictions (b) Selected Model Predictions (c) Highlight False-Negative Regions (d) Final Created Annotation

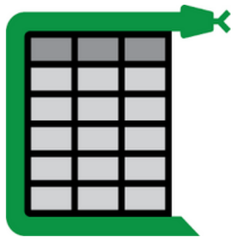
https://arxiv.org/pdf/2010.01762.pdf

Otras utilidades :)

Table 6.1: The highlights of RRE-II coverage (till 11 March, 2010)

State	Date	Halt stations	Halt days	Persons directly reached (in lakh)	Persons trained	Persons counseled	Persons tested for HIV
Delhi	1.12.2009	8	17	1.29	3,665	2,409	1,000
Rajasthan	2.12.2009 to 19.12.2009						
Gujarat	20.12.2009 to 3.1.2010	6	13	6.03	3,810	2,317	1,453
Maharashtra	4.01.2010 to 1.2.2010	13	26	1.27	5,680	9,027	4,153
Karnataka	2.2.2010 to 22.2.2010	11	19	1.80	5,741	3,658	3,183
Kerala	23.2.2010 to 11.3.2010	9	17	1.42	3,559	2,173	855
Total		**47**	**92**	**11.81**	**22,455**	**19,584**	**10,644**

* Includes visitors to train exhibition and those reached through outreach activities



Camelot