



Extracción Estructurada de Documentos con Deep Learning

Dr. Cristian Muñoz Villalobos

Introducción

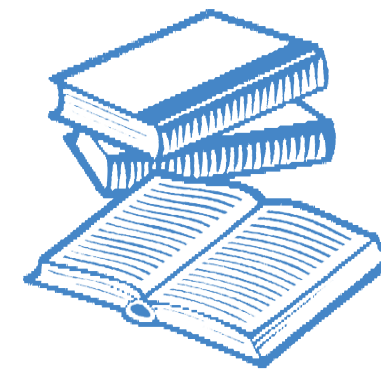
La extracción de documentos estructurados es una tarea importante en muchas organizaciones y áreas de investigación. Encaja como un proceso de organización de datos (Data Curation) donde la recolección y gestión de datos son pasos clave para tener un fácil acceso que permita encontrar y comprender los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



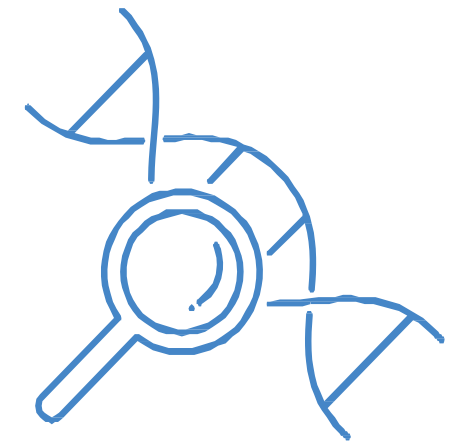
Negocios



Normas



Economía



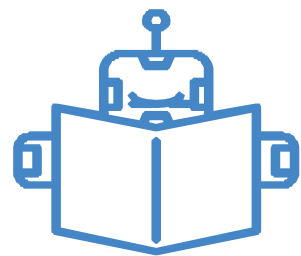
Biomedicina

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.

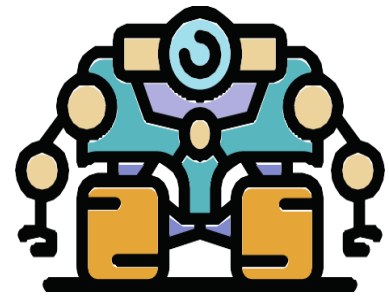


- Podemos dejar que la computadora aprenda a procesar la fecha a partir de ejemplos empíricos, utilizando el aprendizaje profundo.

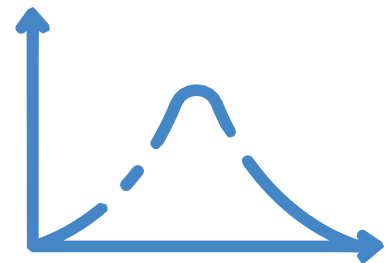
Enfoques basados en reglas

- 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?

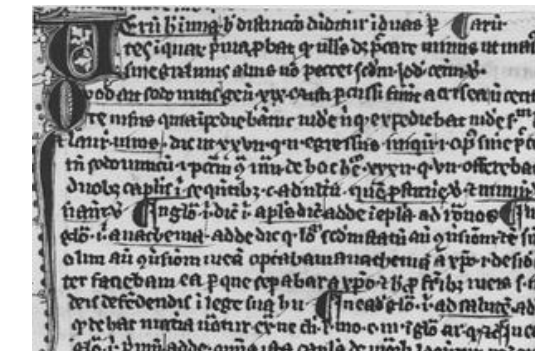
Necesitamos detectar y extraer:

- 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



- Script Type?
- When?
- Where?

- Reconocimiento de emociones en escenas de Comics

Onomatopoeia

Visual appearance of the comic character



Text

Google Vision OCRed text: lund then danny nash screams amp screams forgetting perhaas that the cellar / s apdaf just as teneshad said / t mas .

- Sistemas Visuales Pregunta Respuesta para documentos



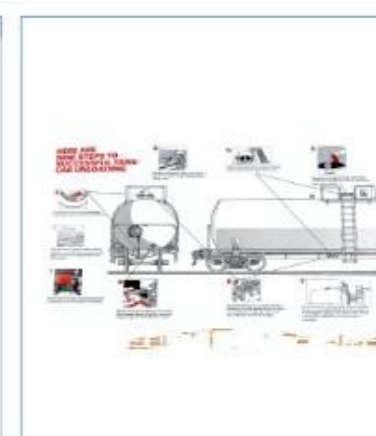
Question: What is the infrastructure support provided by ITC to primary Schools?

Answer: Benches, classrooms, toilets, electrical fixtures, compound walls and gates



Question: What is the Extension Number as per the voucher?

Answer: (910) 741-0673



Question: How many boxed illustrations are there?

Answer: 9



Question: In which years did Anna M. Rivers run for the State senator office?

Positive evidences: 454, 10901
Answers: 2016, 2020

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>

Document
Understanding AI^{BETA}
Use machine learning to unlock insights
from your documents.



<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>

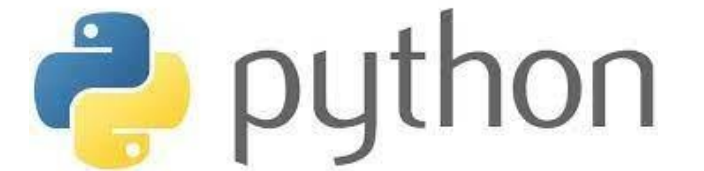
- Desafortunadamente, no existen soluciones listas para usar para todos los tipos de conservación de datos (en CV o NLP), o las soluciones comerciales existentes a menudo no se acercan a una precisión aceptable.
- Una de las plataformas de código abierto más populares para resolver este tipo de problemas fue propuesta por la profesora Melissa Dell de la Universidad de Stanford.



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



Layout Parser: Una solución Open Source



```
>>> 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
```



Figure 7: Annotation Examples in HJDataset. (a) and (b) show two examples for the labeling of main pages. The boxes are colored differently to reflect the layout element categories. Illustrated in (c), the items in each index page row are categorized as title blocks, and the annotations are denser.

over union (IOU) level [0.50:0.95], on the test data. In general, the high mAP values indicate accurate detection of the layout elements. The Faster R-CNN and Mask R-CNN achieve comparable results, better than RetinaNet. Noticeably, the detections for small blocks like title are less precise, and the accuracy drops sharply for the title category. In Figure 8, (a) and (b) illustrate the accurate prediction results of the Faster R-CNN model.

Pre-training for other datasets

We also examine how our dataset can help with a real-world document digitization application. When digitizing new publications, researchers usually do not generate large-scale ground truth data to train their layout analysis models. If they are able to adapt our dataset, or models trained on our dataset, to develop models on their data, they can build their pipelines more efficiently and develop more accurate models. To this end, we conduct two experiments. First, we examine how layout analysis models trained on the main pages can be used for understanding index pages. Moreover, we study how the pre-trained models perform on other historical Japanese documents.

Table 4 compares the performance of five Faster R-CNN models that are trained differently on HJDataset. If the model loads pre-trained weights from HJDataset, it includes information learned from main pages. Models trained over

Table 4: Training Mask R-CNN, the segmentation masks are the quadrilateral regions for each block. Compared to the rectangular bounding boxes, they delineate the text region more accurately.

Table 4: This is a core metric developed for the COCO competition [1] for evaluating the object detection quality.

Table 3: Detection mAP @ IOU [0.50:0.95] of different models for each category on the test set. All values are given as percentages.

Category	Faster R-CNN	Mask R-CNN*	RetinaNet
Page Frame	99.046	99.097	99.038
Row	98.831	98.482	95.067
Title Region	87.571	89.483	69.593
Text Region	94.463	86.798	89.531
Title	65.908	71.517	72.566
Subtitle	84.093	84.174	85.865
Other	44.023	39.849	14.371
mAP	81.991	81.343	75.223

*Training Mask R-CNN, the segmentation masks are the quadrilateral regions for each block. Compared to the rectangular bounding boxes, they delineate the text region more accurately.

```
>>> 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
```



Region Types

- TextRegion
- ImageRegion
- TableRegion

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
```



Region Types

- Text Region
- Title Region
- Title
- Subtitle

Historical Documents

Region Types

- Text
- Title
- Figure
- Table

Paper with Complex Layouts

Lp Layout Parser

```
import layoutparser as lp
image = cv2.imread(...)
model = lp.Detectron2LayoutModel('lp://PrimaLayout/mask_rcnn_R_50_FPN_3x/config')
layout = model.detect(image)
```



Paper with Complex Layouts



Magazine Scans & Websites



Historical Documents

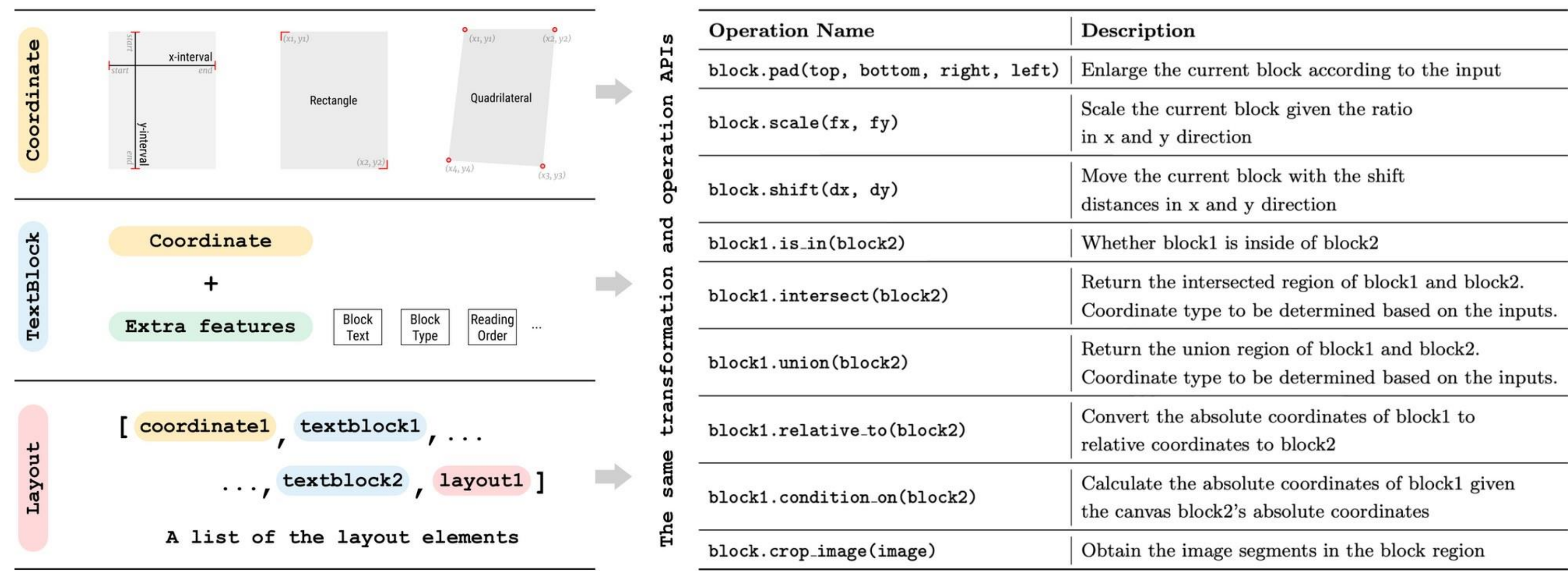
Una riqueza de modelos pre entrenado en conjuntos de datos diferentes



TableBank

Post-Processing

Manipulación y procesamiento de los componentes detectados:



Post-Processing

Visualizar, exportar y almacenar!

Model Customization

Efficient Data Annotation

Customized Model Training

OCR Module

Document Images

Layout Detection Models

Layout Data Structure

Community Platform

DIA Model Hub

DIA Pipeline Sharing

Storage & Visualization

The Core LayoutParser Library

Figure 1: The overall architecture of LayoutParser. For an input document image, the core LayoutParser library provides a set of off-the-shelf tools for layout detection, OCR, visualization, and storage, backed by a carefully designed layout data structure. LayoutParser also supports high level of customization via efficient layout data annotation and model training functions that improves model accuracy on the target samples. The community platform enables the easy share of DIA models and even whole digitization pipelines to promote reusability and reproducibility. A collection of detailed documentations, tutorials and exemplar projects makes LayoutParser easy to learn and use.

Text: LayoutParser is also highly customizable, integrated with functions for layout data annotation and model training. We provide detailed descriptions for each component as follows.

3.1 Layout Detection Models

Text: LayoutParser, a layout model takes a document image as an input and generates a list of rectangular boxes for the target content regions. Different from traditional methods, it relies on deep convolutional neural networks rather than manually curated rules for identifying the content regions. It is formulated as an object detection problem and state of the art models like Fast RCNNs [22] and Mask RCNN [11] are being used. Not only it yields prediction results of high accuracy, but also makes it possible to build a concise while generalized interface for using the layout detection models. In fact, built upon Detectron2 [28], LayoutParser provides a minimal API that one can perform layout detection with only four lines of code in Python:

```
import layoutparser as lp
image = cv2.imread("image_file") # load images
model = lp.Detectron2LayoutModel(
    "lp://PubLayNet/faster_rcnn_R_50_FPN_3x/config")
layout = model.detect(image)
```

Mode I: Showing Layout on the Original Image

Fig. 1: The overall architecture of LayoutParser. For an input document image, the core LayoutParser library provides a set of off-the-shelf tools for layout detection, OCR, visualization, and storage, backed by a carefully designed layout data structure. LayoutParser also supports high level of customization via efficient layout data annotation and model training functions that improves model accuracy on the target samples. The community platform enables the easy share of DIA models and even whole digitization pipelines to promote reusability and reproducibility. A collection of detailed documentations, tutorials and exemplar projects makes LayoutParser easy to learn and use.

LayoutParser is also highly customizable, integrated with functions for layout data annotation and model training. We provide detailed descriptions for each component as follows.

3.1 Layout Detection Models

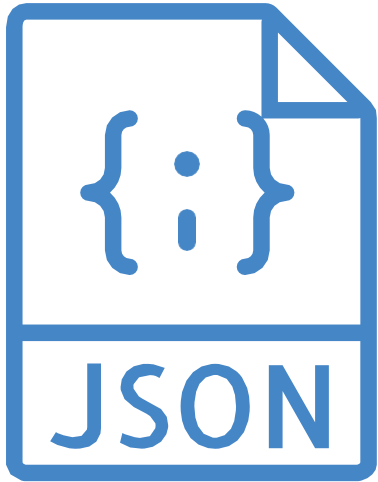
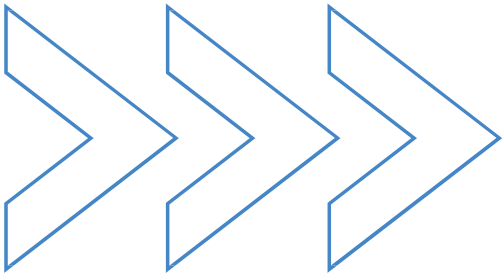
In LayoutParser, a layout model takes a document image as an input and generates a list of rectangular boxes for the target content regions. Different from traditional methods, it relies on deep convolutional neural networks rather than manually curated rules for identifying the content regions. It is formulated as an object detection problem and state of the art models like Fast RCNNs [22] and Mask RCNN [11] are being used. Not only it yields prediction results of high accuracy, but also makes it possible to build a concise while generalized interface for using the layout detection models. In fact, built upon Detectron2 [28], LayoutParser provides a minimal API that one can perform layout detection with only four lines of code in Python:

```
import layoutparser as lp
2 image = cv2.imread("image_file") # load images
3 model = lp.Detectron2LayoutModel(
4     "lp://PubLayNet/faster_rcnn_R_50_FPN_3x/config")
5 layout = model.detect(image)
```

Option 1: Display Token Bounding Box

Option 2: Hide Token Bounding Box

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



Personalización

- Entrenamiento de los modelos.

Layout-Parser/**layout-model-training**



The scripts for training Detectron2-based Layout Models on popular layout analysis datasets

2 Contributors

2 Issues

59 Stars

15 Forks



Layout-Parser/layout-model-training: The scripts for training Detectron2-based Layout Models on popular layout analysis...

The scripts for training Detectron2-based Layout Models on popular layout analysis datasets - GitHub - Layout-Parser/layout-model-training: The scripts for training Detectron2-based Layout Models ...

GitHub

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

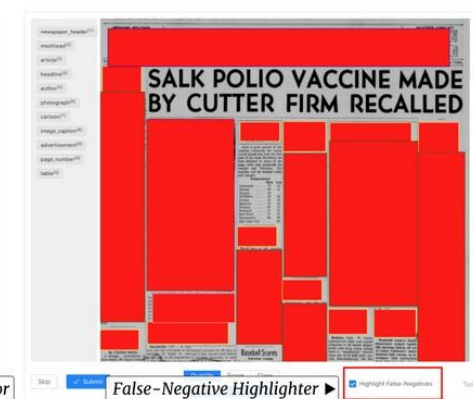
- Anotación de datos eficientes



(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 :)

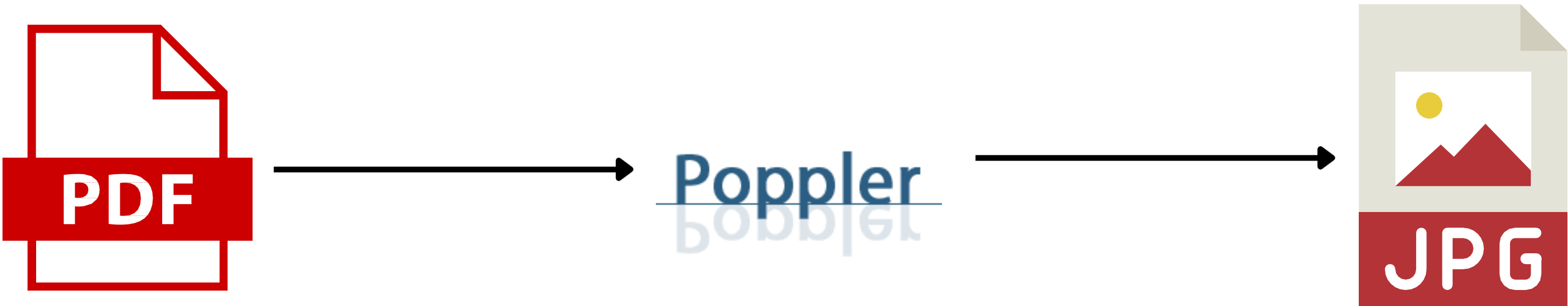


Table 6.1: The highlights of RRE-II coverage (till 11 March, 2010)

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	5,690	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	22,435	19,584	10,644

It includes visitors to state exhibition and those reached through outreach activities

