GIT Link: https://github.com/RajeshBisht28/TableRowColumnPretrained

Google Colab :
https://colab.research.google.com/drive/1miDwYw1LGcC1b6pPf93hDUCS3DYrfLJU#scrollTo=QCgfALxjC09U

Objective: Prediction of Table Row/Column values.

Using Model loaders from the Hugging Face Transformers library.

1: AutoModelForObjectDetection : Model Loader for "microsoft/table-transformer-detection"
2: TableTransformerForObjectDetection : Model Loader for "microsoft/table-structure-recognition-v1.1-all"
3: TrOCRProcessor : Model Loader for "microsoft/trocr-base-printed"

CNN-Deep Learning: Image Processing Pipeline: Using Pretrained Models through Hugging Face

Workflow:
Image Acquisition → Preprocessing → Filtering → Normalization → Transformation → Segmentation → Feature Extraction → Classification/Recognition → Postprocessing → Visualization/Output

---

Overview of an Image Transformation Pipeline

An image transformation pipeline is a sequence of steps or operations applied to images to convert them from one form to another or to extract useful information. It is commonly used in **image processing**, **computer vision**, and **machine learning** tasks.

Pipeline Steps

1. 🖼️ **Image Acquisition**
   Capturing or obtaining the image from a source such as a camera, scanner, or an image file.

2. 🛠️ **Preprocessing**
   Performing initial adjustments to the image, such as resizing, cropping, denoising, and correcting for brightness and contrast.
3. 🔍 **Filtering**
   Applying filters to enhance or detect specific features like edge detection, blurring, or sharpening.
4. 📊 **Normalization**
   Scaling image pixel values to a specific range (e.g., 0 to 1) to standardize inputs.
5. ⚙️ **Transformation**
   Applying geometric changes such as rotation, scaling, and perspective adjustments.
6. 🧩 **Segmentation**
   Dividing the image into meaningful regions or objects, crucial for object detection tasks.
7. 🌟 **Feature Extraction**
   Identifying key features like edges, corners, or textures for model inputs.
8. 🗂️ **Classification/Recognition**
   Using machine learning or deep learning models to classify images or detect objects.
9. ✨ **Postprocessing**
   Refining outputs, such as smoothing or merging results.
10. 📈 **Visualization/Output**
    Displaying or saving the processed image with annotations or extracted data.

```
####  load a version of the model revision="no_timm"
```

```
Start coding or generate with AI.
```

```
import warnings
warnings.filterwarnings("ignore")
```

```
from transformers import AutoModelForObjectDetection
table_detection_model = AutoModelForObjectDetection.from_pretrained("microsoft/table-transformer-detection", revision="no_t
```

```
config.json:   0%|          | 0.00/76.5k [00:00<?, ?B/s]
model.safetensors:   0%|          | 0.00/115M [00:00<?, ?B/s]
```

```
#### Image path which contain Table...
table_image_path = r"E:\TEST_FILES\image_tables\page2.png"
table_image_path = r"E:\TEST_FILES\image_data\Table.png"
```

Check Compute Unified Device Architecture (CUDA) from NVIDIA that accelerates machine learning and data science on GPUs.

Check if CUDA is available and set the device otherwise CPU.

```
import torch
device = "cuda" if torch.cuda.is_available() else "cpu"
table_detection_model.to(device)
print("")
```

Load image from local path, For Display do some scaling.

```
from PIL import Image
from huggingface_hub import hf_hub_download

image = Image.open(table_image_path).convert("RGB")
# let's display it a bit smaller
width, height = image.size
display(image.resize((int(0.6*width), (int(0.6*height)))))
```

Roll	Name	Opening	Purchase	Number
P101	Laptop	50	20	10
P102	Monitor	40	15	55
P103	Keyboard	60	25	15
P104	Headphones	30	10	66
P105	Smartphone	70	30	20
P106	Tablet	45	18	88
P107	Router	55	22	12
P108	External Hard Drive	25	12	99
P109	Wireless Earbuds	35	15	33
P110	Webcam	40	20	10

Double-click (or enter) to edit

```
from torchvision import transforms
class ImageScalerResizer(object):
    def __init__(self, max_size=800):
        self.max_size = max_size

    def __call__(self, image):
        w, h = image.size
        scale = self.max_size / max(width, height)
        resized_image = image.resize((int(round(scale*w)), int(round(scale*h))))
        return resized_image

detection_transform = transforms.Compose([
    ImageScalerResizer(800),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])
transformed_pixels = detection_transform(image).unsqueeze(0)
transformed_pixels = transformed_pixels.to(device)
print("========== Information of model transform =============")

# Get the dimensions
batch_size, channels, height, width = transformed_pixels.shape

# Display the dimensions values...
print(f"Batch size: {batch_size}")
```

```
print(f"Number of channels: {channels}")
print(f"Height: {height}")
print(f"Width: {width}")
```

```
========== Information of model transform =============
Batch size: 1
Number of channels: 3
Height: 396
Width: 800
```

using CNN: convolutional layers : predict outcomes => using Pytorch library and table detection model

- **There is ignore: Gradient calculation during prediction outcomes.**
- **outcomes: Context of neural networks, to refer to the raw, unnormalized output of a model.**

```
import torch
with torch.no_grad():
  outputs = table_detection_model(transformed_pixels)
outputs.logits.shape
```

```
torch.Size([1, 15, 3])
```

Double-click (or enter) to edit

outputs.logits.shape : Refers : torch.Size([1, 15, 3]) ==> 1=batch size, 15=number of objects or detections by model and 3=number of classes or categories

Extracts the predicted labels, scores, and bounding boxes from the model's output.

-**Applies softmax to the logits to get the probabilities for each class, and then takes the class with the highest probability.**

-**Rescales the bounding boxes to match the image size.**

-**Filters out predictions labeled as "no object".**

-**Returns a list of detected objects, each with its label, score, and bounding box coordinates.**

```
# update id2label to include "no object"
id2label = table_detection_model.config.id2label
id2label[len(table_detection_model.config.id2label)] = "no object"

#  post-processing for bounding boxes values
def box_cxcywh_to_xyxy(x):
    x_c, y_c, w, h = x.unbind(-1)
    b = [(x_c - 0.5 * w), (y_c - 0.5 * h), (x_c + 0.5 * w), (y_c + 0.5 * h)]
    return torch.stack(b, dim=1)


def rescale_bboxes(out_bbox, size):
    img_w, img_h = size
    b = box_cxcywh_to_xyxy(out_bbox)
    b = b * torch.tensor([img_w, img_h, img_w, img_h], dtype=torch.float32)
    return b

def fetch_label_bbx_score(outputs, img_size, id2label):
    m = outputs.logits.softmax(-1).max(-1)
    pred_labels = list(m.indices.detach().cpu().numpy())[0]
    pred_scores = list(m.values.detach().cpu().numpy())[0]
    pred_bboxes = outputs['pred_boxes'].detach().cpu()[0]
    pred_bboxes = [elem.tolist() for elem in rescale_bboxes(pred_bboxes, img_size)]

    tableinfo = []
    for label, score, bbox in zip(pred_labels, pred_scores, pred_bboxes):
        class_label = id2label[int(label)]
        if not class_label == 'no object':
            tableinfo.append({'label': class_label, 'score': float(score),
                              'bbox': [float(elem) for elem in bbox]})

    return tableinfo


detected_table_boundry = fetch_label_bbx_score(outputs, image.size, id2label)
```

```
print(detected_table_boundry)
```

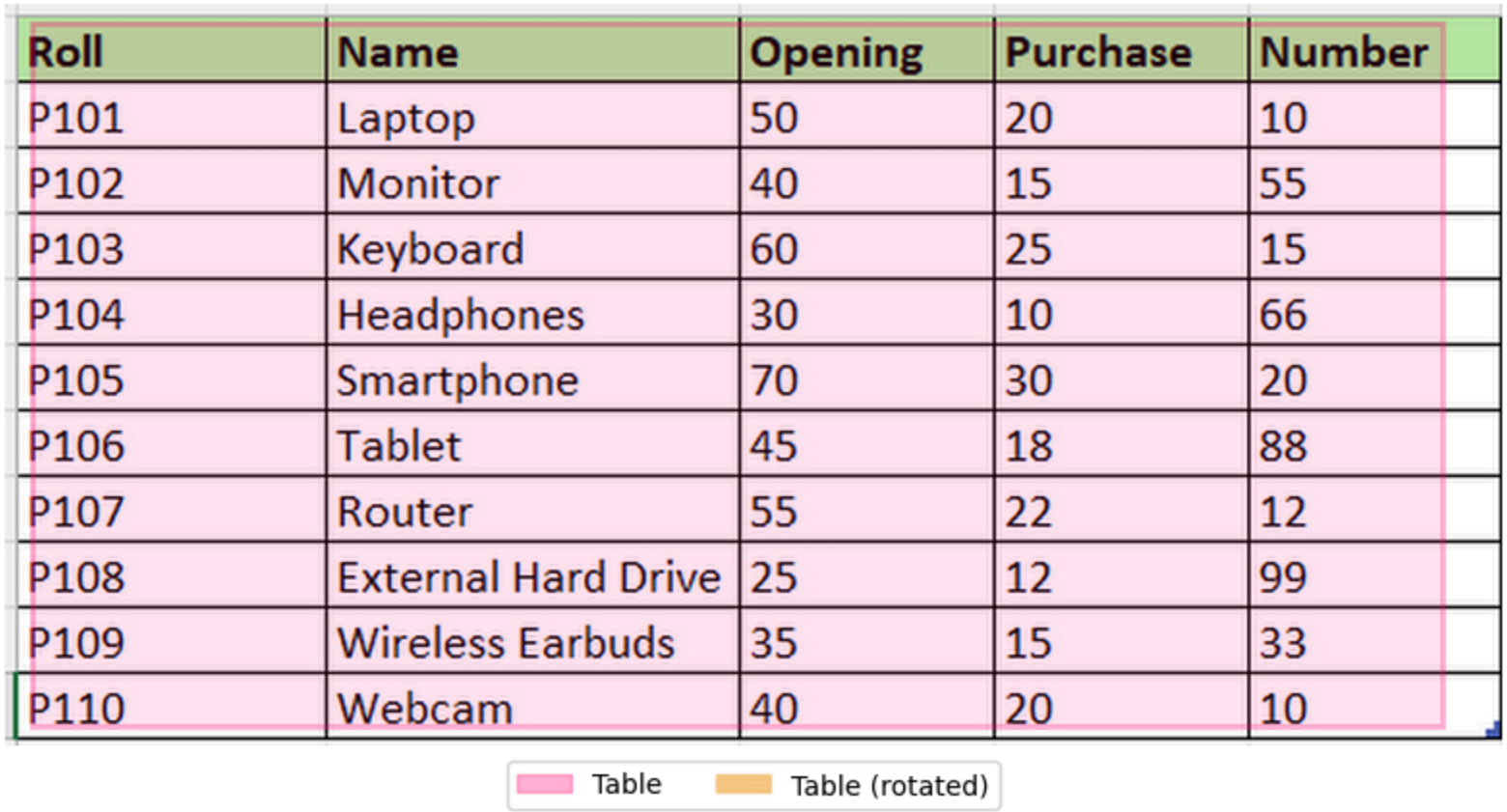
```
[{'label': 'table', 'score': 0.9997441172599792, 'bbox': [9.19443416595459, 6.62215518951416, 524.3612670898438, 263.35
```

Visualize: Detected image. (Detected by table transform model)

```
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from matplotlib.patches import Patch

def fig2img(fig):
    """Convert a Matplotlib figure to a PIL Image and return it"""
    import io
    buf = io.BytesIO()
    fig.savefig(buf)
    buf.seek(0)
    img = Image.open(buf)
    return img


def visualize_detected_tables(img, det_tables, out_path=None):
    plt.imshow(img, interpolation="lanczos")
    fig = plt.gcf()
    fig.set_size_inches(20, 20)
    ax = plt.gca()

    for det_table in det_tables:
        bbox = det_table['bbox']
        ### increases the x-coordinate of the bottom-right corner by 15, effectively expanding the bounding box horizontall
        bbox[2] = bbox[2] ##+15
        if det_table['label'] == 'table':
            facecolor = (1, 0, 0.45)
            edgecolor = (1, 0, 0.45)
            alpha = 0.3
            linewidth = 2
```

```
        hatch= '~~~~~~'
    elif det_table['label'] == 'table rotated':
        facecolor = (0.95, 0.6, 0.1)
        edgecolor = (0.95, 0.6, 0.1)
        alpha = 0.3
        linewidth = 2
        hatch= '******'
    else:
        continue

    rect = patches.Rectangle(bbox[:2], bbox[2]-bbox[0], bbox[3]-bbox[1], linewidth=linewidth,
                             edgecolor='none',facecolor=facecolor, alpha=0.1)
    ax.add_patch(rect)
    rect = patches.Rectangle(bbox[:2], bbox[2]-bbox[0], bbox[3]-bbox[1], linewidth=linewidth,
                             edgecolor=edgecolor,facecolor='none',linestyle='-', alpha=alpha)
    ax.add_patch(rect)
    rect = patches.Rectangle(bbox[:2], bbox[2]-bbox[0], bbox[3]-bbox[1], linewidth=0,
                             edgecolor=edgecolor,facecolor='none',linestyle='-', hatch=hatch, alpha=0.2)
    ax.add_patch(rect)

plt.xticks([], [])
plt.yticks([], [])

legend_elements = [Patch(facecolor=(1, 0, 0.45), edgecolor=(1, 0, 0.45),
                         label='Table', hatch='~~~~~~', alpha=0.3),
                   Patch(facecolor=(0.95, 0.6, 0.1), edgecolor=(0.95, 0.6, 0.1),
                         label='Table (rotated)', hatch='******', alpha=0.3)]
plt.legend(handles=legend_elements, bbox_to_anchor=(0.5, -0.02), loc='upper center', borderaxespad=0,
               fontsize=10, ncol=2)
plt.gcf().set_size_inches(10, 10)
plt.axis('off')

if out_path is not None:
  plt.savefig(out_path, bbox_inches='tight', dpi=150)

return fig
```

```
fig = visualize_detected_tables(image, detected_table_boundry)
```




```
visualized_image = fig2img(fig)
```

Crop table with some padding: using boundig boxes, label =>> For furthur processing

-Table Transformer (TATR) model, padding suggest to adding extra space around the detected table region in an image.

-padding helps the model better recognize and process the table structure.

```
def crops_table_object(img, tokens, objects, class_thresholds, padding=10):
    table_object = []
```

```
    for obj in objects:
        if obj['score'] < class_thresholds[obj['label']]:
            continue

        cropped_table = {}

        bbox = obj['bbox']
        bbox = [bbox[0]-padding, bbox[1]-padding, bbox[2]+padding, bbox[3]+padding]

        cropped_img = img.crop(bbox)

        table_tokens = [token for token in tokens if iob(token['bbox'], bbox) >= 0.5]
        for token in table_tokens:
            token['bbox'] = [token['bbox'][0]-bbox[0],
                             token['bbox'][1]-bbox[1],
                             token['bbox'][2]-bbox[0],
                             token['bbox'][3]-bbox[1]]

        # If table is predicted to be rotated, rotate cropped image and tokens/words:
        if obj['label'] == 'table rotated':
            cropped_img = cropped_img.rotate(270, expand=True)
            for token in table_tokens:
                bbox = token['bbox']
                bbox = [cropped_img.size[0]-bbox[3]-1,
                        bbox[0],
                        cropped_img.size[0]-bbox[1]-1,
                        bbox[2]]
                token['bbox'] = bbox

        cropped_table['image'] = cropped_img
        cropped_table['tokens'] = table_tokens

        table_object.append(cropped_table)

    return table_object


detected_table_boundry
```

```
[{'label': 'table',
  'score': 0.9997441172599792,
  'bbox': [9.19443416595459,
   6.62215518951416,
   524.3612670898438,
   263.3559875488281]}]
```

```
tokens = []
detection_class_thresholds = {
    "table": 0.5,
    "table rotated": 0.5,
    "no object": 10
}
crop_padding = 10

tables_crops = crops_table_object(image, tokens, detected_table_boundry, detection_class_thresholds, padding=0)
cropped_table = tables_crops[0]['image'].convert("RGB")
cropped_table
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[4], line 9
      2 detection_class_thresholds = {
      3     "table": 0.5,
      4     "table rotated": 0.5,
      5     "no object": 10
      6 }
      7 crop_padding = 10
----> 9 tables_crops = crops_table_object(image, tokens, detected_table_boundry, detection_class_thresholds,
padding=0)
     10 cropped_table = tables_crops[0]['image'].convert("RGB")
     11 cropped_table

NameError: name 'image' is not defined
```

```
cropped_table.save("table2.jpg")
```

Load structure recognition model

Next, we load a Table Transformer pre-trained for table structure recognition.

```
from transformers import TableTransformerForObjectDetection

# new v1.1 checkpoints require no timm anymore
structure_model = TableTransformerForObjectDetection.from_pretrained("microsoft/table-structure-recognition-v1.1-all")
structure_model.to(device)
print("")
```

Perform: Data preprocessing pipeline,

```
structure_transform = transforms.Compose([
    ImageScalerResizer(1000),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])
transformed_pixels = structure_transform(cropped_table).unsqueeze(0)
transformed_pixels = transformed_pixels.to(device)
print(transformed_pixels.shape)
```

Start coding or generate with AI.

```
torch.Size([1, 3, 320, 644])
```

Prepare the cropped table image for the model, and perform a forward pass.

```
# forward pass
with torch.no_grad():
  outputs = structure_model(transformed_pixels)
```

Next, we get the predicted detections.

```
# update id2label to include "no object"
structure_id2label = structure_model.config.id2label
structure_id2label[len(structure_id2label)] = "no object"

cells = fetch_label_bbx_score(outputs, cropped_table.size, structure_id2label)
###>>>print(cells)
```

Visualize cells

We can visualize all recognized cells using PIL's ImageDraw module.

```
from PIL import ImageDraw

cropped_table_visualized = cropped_table.copy()
draw = ImageDraw.Draw(cropped_table_visualized)
print(f"cell counts: {len(cells)}")

for cell in cells:
       draw.rectangle(cell["bbox"], outline="green")

cropped_table_visualized
```

```
cell counts: 18
```

Roll		Name	Opening		Purchase		Number
P101		Laptop	50		20		10
P102		Monitor	40		15		55
P103		Keyboard	60		25		15
P104		Headphones	30		10		66
P105		Smartphone	70		30		20
P106		Tablet	45		18		88
P107		Router	55		22		12
P108		External Hard Drive	25		12		99
P109		Wireless Earbuds	35		15		33
P110		Webcam	40		20		10

An alternative way of plotting is to select one class to visualize, like "table row":

```
def plot_results(cells, class_to_visualize):
    if class_to_visualize not in structure_model.config.id2label.values():
      raise ValueError("Class should be one of the available classes")

    plt.figure(figsize=(16,10))
    plt.imshow(cropped_table)
    ax = plt.gca()

    for cell in cells:
        score = cell["score"]
        bbox = cell["bbox"]
        label = cell["label"]
        ## >>> print(f"label: {label}")
        if label == class_to_visualize:
          xmin, ymin, xmax, ymax = tuple(bbox)

          ax.add_patch(plt.Rectangle((xmin, ymin), xmax - xmin, ymax - ymin, fill=False, color="red", linewidth=3))
          text = f'{cell["label"]}: {score:0.2f}'
          ax.text(xmin, ymin, text, fontsize=15,
```

```
                bbox=dict(facecolor='white', alpha=0.5))
        plt.axis('off')


plot_results(cells, class_to_visualize="table row")
```

table row: 1.00

Roll	Name	Opening	Purchase	Number
P101	Laptop	50	20	10
P102	Monitor	40	15	55
P103	Keyboard	60	25	15
P104	Headphones	30	10	66
P105	Smartphone	70	30	20
P106	Tablet	45	18	88
P107	Router	55	22	12
P108	External Hard Drive	25	12	99
P109	Wireless Earbuds	35	15	33
P110	Webcam	40	20	10

Courtesy: Copilot / GeminiAi / ChatGPT : Apply OCR row by row : ROI (Region of Interaction).

Identify Intersections And Calculate Intersection Area:

```
def get_cell_coordinates_by_row(table_data):
    # Extract rows and columns
    rows = [entry for entry in table_data if entry['label'] == 'table row']
    columns = [entry for entry in table_data if entry['label'] == 'table column']

    # Sort rows and columns by their Y and X coordinates, respectively
    rows.sort(key=lambda x: x['bbox'][1])
    columns.sort(key=lambda x: x['bbox'][0])

    # Function to find cell coordinates
    def find_cell_coordinates(row, column):
        cell_bbox = [column['bbox'][0], row['bbox'][1], column['bbox'][2], row['bbox'][3]]
        return cell_bbox

    # Generate cell coordinates and count cells in each row
    cell_coordinates = []

    for row in rows:
        row_cells = []
        for column in columns:
            cell_bbox = find_cell_coordinates(row, column)
            row_cells.append({'column': column['bbox'], 'cell': cell_bbox})

        # Sort cells in the row by X coordinate
        row_cells.sort(key=lambda x: x['column'][0])

        # Append row information to cell_coordinates
        cell_coordinates.append({'row': row['bbox'], 'cells': row_cells, 'cell_count': len(row_cells)})

    # Sort rows from top to bottom
    cell_coordinates.sort(key=lambda x: x['row'][1])

    return cell_coordinates

cell_coordinates = get_cell_coordinates_by_row(cells)
```

```
len(cell_coordinates)
```

```
11
```

```
len(cell_coordinates[0]["cells"])
```

```
5
```

Alternate for OCR : model : microsoft/trocr-base-printed

EasyOCr is primary: if EasyOCr unable to read then use Trocr Model.

```
from transformers import TrOCRProcessor, VisionEncoderDecoderModel
from PIL import Image
import torch

ocr_processor = TrOCRProcessor.from_pretrained('microsoft/trocr-base-printed')
ocr_model = VisionEncoderDecoderModel.from_pretrained('microsoft/trocr-base-printed')

def apply_trocr(cell_image, ocr_processor=ocr_processor, ocr_model=ocr_model ):
    pix_vals = ocr_processor(images=cell_image, return_tensors="pt").pixel_values
    gen_ids = ocr_model.generate(pix_vals)
    gen_text = ocr_processor.batch_decode(gen_ids, skip_special_tokens=True)[0]
    return gen_text
```

```
Config of the encoder: <class 'transformers.models.vit.modeling_vit.ViTModel'> is overwritten by shared encoder config:
  "attention_probs_dropout_prob": 0.0,
  "encoder_stride": 16,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.0,
  "hidden_size": 768,
  "image_size": 384,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "model_type": "vit",
```

```
  "num_attention_heads": 12,
  "num_channels": 3,
  "num_hidden_layers": 12,
  "patch_size": 16,
  "qkv_bias": false,
  "transformers_version": "4.48.0.dev0"
}

Config of the decoder: <class 'transformers.models.trocr.modeling_trocr.TrOCRForCausalLM'> is overwritten by shared dec
  "activation_dropout": 0.0,
  "activation_function": "gelu",
  "add_cross_attention": true,
  "attention_dropout": 0.0,
  "bos_token_id": 0,
  "classifier_dropout": 0.0,
  "cross_attention_hidden_size": 768,
  "d_model": 1024,
  "decoder_attention_heads": 16,
  "decoder_ffn_dim": 4096,
  "decoder_layerdrop": 0.0,
  "decoder_layers": 12,
  "decoder_start_token_id": 2,
  "dropout": 0.1,
  "eos_token_id": 2,
  "init_std": 0.02,
  "is_decoder": true,
  "layernorm_embedding": true,
  "max_position_embeddings": 512,
  "model_type": "trocr",
  "pad_token_id": 1,
  "scale_embedding": false,
  "transformers_version": "4.48.0.dev0",
  "use_cache": false,
  "use_learned_position_embeddings": true,
  "vocab_size": 50265
}

Some weights of VisionEncoderDecoderModel were not initialized from the model checkpoint at microsoft/trocr-base-printe
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

Start coding or generate with AI.

Using EasyOcr : Read and Collect Text. (Alternate TrOCr pretrained model also using.)

```
import numpy as np
import csv
import easyocr
from tqdm.auto import tqdm

reader = easyocr.Reader(['en'])

def apply_ocr(cell_coordinates):
    # let's OCR row by row
    data = dict()
    max_num_columns = 0
    indx = 0
    for idx, row in enumerate(tqdm(cell_coordinates)):
      row_text = []
      for cell in row["cells"]:
        indx = indx + 1
        # crop cell out of image
        cell_image = np.array(cropped_table.crop(cell["cell"]))
        # apply OCR
        result = reader.readtext(np.array(cell_image))
        if len(result) > 0:
          # print([x[1] for x in list(result)])
          text = " ".join([x[1] for x in result])
          row_text.append(text)
        else:  ## >> testing-comment
          image_to_ocr = Image.fromarray(cell_image)
          text = apply_trocr(image_to_ocr)
          row_text.append(text)

      if len(row_text) > max_num_columns:
          max_num_columns = len(row_text)

      data[idx] = row_text
```

```
    print("Max number of columns:", max_num_columns)

    # pad rows which don't have max_num_columns elements
    # to make sure all rows have the same number of columns
    for row, row_data in data.copy().items():
        if len(row_data) != max_num_columns:
          row_data = row_data + ["" for _ in range(max_num_columns - len(row_data))]
        data[row] = row_data

    return data

data = apply_ocr(cell_coordinates)

for row, row_data in data.items():
    print(row_data)
```

```
Neither CUDA nor MPS are available - defaulting to CPU. Note: This module is much faster with a GPU.
  0%|          | 0/11 [00:00<?, ?it/s]
Max number of columns: 5
['Roll', 'Name', 'Opening', 'Purchase', 'Number']
['P101', 'Laptop', '50', '20', '10']
['P102', 'Monitor', '40', '15', '55']
['P103', 'Kevboard', '60', '25', '15']
['P104', 'Headphones', '30', '10', '66']
['P105', 'Smartphone', '70', '30', '20']
['P106', 'Tablet', '45', '18', '88']
['P107', 'Router', '55', '22', '12']
['P108', 'External Hard Drive', '25', '12', '99']
['P10g', 'Wireless Earbuds', '35', '15', '33']
['P11O', 'Webcam', '40', '20', '10']
```

Save as CSV

We end up with a CSV file containing the data.

```
import csv
try:
    if os.path.exists("output.csv"):
     os.remove("output.csv")
except:
    pass

with open('output.csv','w') as result_file:
    wr = csv.writer(result_file, dialect='excel')

    for row, row_text in data.items():
      wr.writerow(row_text)


import pandas as pd

df = pd.read_csv("output.csv")
df.head()
```

	Roll	Name	Opening	Purchase	Number
0	P101	Laptop	50	20	10
1	P102	Monitor	40	15	55
2	P103	Kevboard	60	25	15
3	P104	Headphones	30	10	66
4	P105	Smartphone	70	30	20

```
pip install --upgrade jupyter nbconvert
```

```
Requirement already satisfied: jupyter in c:\programdata\anaconda3\lib\site-packages (1.0.0)
Collecting jupyter
  Downloading jupyter-1.1.1-py2.py3-none-any.whl.metadata (2.0 kB)
Requirement already satisfied: nbconvert in c:\programdata\anaconda3\lib\site-packages (7.16.4)
Collecting nbconvert
  Downloading nbconvert-7.16.6-py3-none-any.whl.metadata (8.5 kB)
Requirement already satisfied: notebook in c:\programdata\anaconda3\lib\site-packages (from jupyter) (7.2.2)
```

```
Requirement already satisfied: jupyter-console in c:\programdata\anaconda3\lib\site-packages (from jupyter) (6.6.3)
Requirement already satisfied: ipykernel in c:\programdata\anaconda3\lib\site-packages (from jupyter) (6.28.0)
Requirement already satisfied: ipywidgets in c:\programdata\anaconda3\lib\site-packages (from jupyter) (7.8.1)
Requirement already satisfied: jupyterlab in c:\programdata\anaconda3\lib\site-packages (from jupyter) (4.2.5)
Requirement already satisfied: beautifulsoup4 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (4.12.3
Requirement already satisfied: bleach!=5.0.0 in c:\programdata\anaconda3\lib\site-packages (from bleach[css]!=5.0.0-
Requirement already satisfied: defusedxml in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (0.7.1)
Requirement already satisfied: jinja2>=3.0 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (3.1.4)
Requirement already satisfied: jupyter-core>=4.7 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (5.7
Requirement already satisfied: jupyterlab-pygments in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (0
Requirement already satisfied: markupsafe>=2.0 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (2.1.3
Requirement already satisfied: mistune<4,>=2.0.3 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (2.0
Requirement already satisfied: nbclient>=0.5.0 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (0.8.0
Requirement already satisfied: nbformat>=5.7 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (5.10.4)
Requirement already satisfied: packaging in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (24.1)
Requirement already satisfied: pandocfilters>=1.4.1 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (
Requirement already satisfied: pygments>=2.4.1 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (2.15.
Requirement already satisfied: traitlets>=5.1 in c:\programdata\anaconda3\lib\site-packages (from nbconvert) (5.14.3
Requirement already satisfied: six>=1.9.0 in c:\programdata\anaconda3\lib\site-packages (from bleach!=5.0.0->bleach[
Requirement already satisfied: webencodings in c:\programdata\anaconda3\lib\site-packages (from bleach!=5.0.0->bleac
Requirement already satisfied: platformdirs>=2.5 in c:\programdata\anaconda3\lib\site-packages (from jupyter-core>=4
Requirement already satisfied: pywin32>=300 in c:\programdata\anaconda3\lib\site-packages (from jupyter-core>=4.7->n
Requirement already satisfied: jupyter-client>=6.1.12 in c:\programdata\anaconda3\lib\site-packages (from nbclient>=
Requirement already satisfied: fastjsonschema>=2.15 in c:\programdata\anaconda3\lib\site-packages (from nbformat>=5.
Requirement already satisfied: jsonschema>=2.6 in c:\programdata\anaconda3\lib\site-packages (from nbformat>=5.7->nb
Requirement already satisfied: soupsieve>1.2 in c:\programdata\anaconda3\lib\site-packages (from beautifulsoup4->nbc
Requirement already satisfied: comm>=0.1.1 in c:\programdata\anaconda3\lib\site-packages (from ipykernel->jupyter) (
Requirement already satisfied: debugpy>=1.6.5 in c:\programdata\anaconda3\lib\site-packages (from ipykernel->jupyter
Requirement already satisfied: ipython>=7.23.1 in c:\programdata\anaconda3\lib\site-packages (from ipykernel->jupyte
Requirement already satisfied: matplotlib-inline>=0.1 in c:\programdata\anaconda3\lib\site-packages (from ipykernel-
Requirement already satisfied: nest-asyncio in c:\programdata\anaconda3\lib\site-packages (from ipykernel->jupyter)
Requirement already satisfied: psutil in c:\programdata\anaconda3\lib\site-packages (from ipykernel->jupyter) (5.9.0
Requirement already satisfied: pyzmq>=24 in c:\programdata\anaconda3\lib\site-packages (from ipykernel->jupyter) (25
Requirement already satisfied: tornado>=6.1 in c:\programdata\anaconda3\lib\site-packages (from ipykernel->jupyter)
Requirement already satisfied: ipython-genutils~=0.2.0 in c:\programdata\anaconda3\lib\site-packages (from ipywidget
Requirement already satisfied: widgetsnbextension~=3.6.6 in c:\programdata\anaconda3\lib\site-packages (from ipywidg
Requirement already satisfied: jupyterlab-widgets<3,>=1.0.0 in c:\programdata\anaconda3\lib\site-packages (from ipyw
Requirement already satisfied: prompt-toolkit>=3.0.30 in c:\programdata\anaconda3\lib\site-packages (from jupyter-co
Requirement already satisfied: async-lru>=1.0.0 in c:\programdata\anaconda3\lib\site-packages (from jupyterlab->jupy
Requirement already satisfied: httpx>=0.25.0 in c:\programdata\anaconda3\lib\site-packages (from jupyterlab->jupyter
Requirement already satisfied: jupyter-lsp>=2.0.0 in c:\programdata\anaconda3\lib\site-packages (from jupyterlab->ju
Requirement already satisfied: jupyter-server<3,>=2.4.0 in c:\programdata\anaconda3\lib\site-packages (from jupyterl
```

```
Requirement already satisfied: jupyterlab-server<3,>=2.27.1 in c:\programdata\anaconda3\lib\site-packages (from jupy
Requirement already satisfied: notebook-shim>=0.2 in c:\programdata\anaconda3\lib\site-packages (from jupyterlab->ju
Requirement already satisfied: setuptools>=40.1.0 in c:\programdata\anaconda3\lib\site-packages (from jupyterlab->ju
Requirement already satisfied: anyio in c:\programdata\anaconda3\lib\site-packages (from httpx>=0.25.0->jupyterlab->
Requirement already satisfied: certifi in c:\programdata\anaconda3\lib\site-packages (from httpx>=0.25.0->jupyterlab
Requirement already satisfied: httpcore==1.* in c:\programdata\anaconda3\lib\site-packages (from httpx>=0.25.0->jupy
Requirement already satisfied: idna in c:\programdata\anaconda3\lib\site-packages (from httpx>=0.25.0->jupyterlab->j
```

```
pip install -U jupyter
```

```
Requirement already satisfied: jupyter in c:\programdata\anaconda3\lib\site-packages (1.1.1)
Requirement already satisfied: notebook in c:\programdata\anaconda3\lib\site-packages (from jupyter) (7.2.2)
Requirement already satisfied: jupyter-console in c:\programdata\anaconda3\lib\site-packages (from jupyter) (6.6.3)
Requirement already satisfied: nbconvert in c:\programdata\anaconda3\lib\site-packages (from jupyter) (7.16.6)
Requirement already satisfied: ipykernel in c:\programdata\anaconda3\lib\site-packages (from jupyter) (6.28.0)
Requirement already satisfied: ipywidgets in c:\programdata\anaconda3\lib\site-packages (from jupyter) (7.8.1)
Requirement already satisfied: jupyterlab in c:\programdata\anaconda3\lib\site-packages (from jupyter) (4.2.5)
Requirement already satisfied: comm>=0.1.1 in c:\programdata\anaconda3\lib\site-packages (from ipykernel->jupyter) (
Requirement already satisfied: debugpy>=1.6.5 in c:\programdata\anaconda3\lib\site-packages (from ipykernel->jupyter
Requirement already satisfied: ipython>=7.23.1 in c:\programdata\anaconda3\lib\site-packages (from ipykernel->jupyte
Requirement already satisfied: jupyter-client>=6.1.12 in c:\programdata\anaconda3\lib\site-packages (from ipykernel-
Requirement already satisfied: jupyter-core!=5.0.*,>=4.12 in c:\programdata\anaconda3\lib\site-packages (from ipyker
Requirement already satisfied: matplotlib-inline>=0.1 in c:\programdata\anaconda3\lib\site-packages (from ipykernel-
Requirement already satisfied: nest-asyncio in c:\programdata\anaconda3\lib\site-packages (from ipykernel->jupyter)
Requirement already satisfied: packaging in c:\programdata\anaconda3\lib\site-packages (from ipykernel->jupyter) (24
Requirement already satisfied: psutil in c:\programdata\anaconda3\lib\site-packages (from ipykernel->jupyter) (5.9.0
Requirement already satisfied: pyzmq>=24 in c:\programdata\anaconda3\lib\site-packages (from ipykernel->jupyter) (25
Requirement already satisfied: tornado>=6.1 in c:\programdata\anaconda3\lib\site-packages (from ipykernel->jupyter)
Requirement already satisfied: traitlets>=5.4.0 in c:\programdata\anaconda3\lib\site-packages (from ipykernel->jupyt
Requirement already satisfied: ipython-genutils~=0.2.0 in c:\programdata\anaconda3\lib\site-packages (from ipywidget
```