Image_Create_Modified

July 3, 2022

0.1 Importing libraries

[1]:

```
from io import StringIO
import json
import os

import cv2
import dash
from dash import dash_table
from dash import dcc
from dash import html
from dash.dependencies import Input, Output
import pandas as pd
from PIL import Image, ImageEnhance, ImageFilter
import plotly.express as px
import plotly.graph_objects as go
import pytesseract
from pytesseract import Output as Output1
from skimage import data
```

0.2 Setting default display for pandas Dataframes

[2]:

```
pd.set_option("display.max_columns", None)
pd.set_option("display.max_rows", None)
```

0.3 Reading image

[3]:

```
fileName = os.path.abspath(".\\ADMIN1.jpg")
img = cv2.imread(fileName)

img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

0.4 Applying Tesseract OCR to image

[5]:

```
pytesseract.pytesseract.tesseract_cmd = r'C:\Program␣
 ↪Files\Tesseract-OCR\tesseract'
d = pytesseract.image_to_data(img, output_type=Output1.DICT)
```

0.5 Processing the DataFrame with the dictionary source

[6]:

```
dfCoord = pd.DataFrame.from_dict(d)
dfCoord = dfCoord[dfCoord["conf"] != "-1"]
dfCoord = dfCoord.drop(["level", "page_num"], axis=1)
dfCoord = dfCoord[dfCoord["text"].apply(lambda x: x.strip()) != ""]
dfCoord = dfCoord.reset_index(drop=True)
```

[15]:

```
# dfCoord
```

0.6 Build the requested structure for the data

[9]:

```
prev_row = dfCoord.iloc[0]["word_num"]
line = dfCoord.iloc[0]["text"] + " "
min_x, min_y = dfCoord.iloc[0]["left"], dfCoord.iloc[0]["top"]
max_x, max_y = (
    dfCoord.iloc[0]["left"] + dfCoord.iloc[0]["width"],
    dfCoord.iloc[0]["top"] + dfCoord.iloc[0]["height"],
)
pre_row = dfCoord.iloc[0]["left"]

preconfig = (
    True,
    "x",
    "y",
    "above",
    1,
    {"color": "red", "width": 1, "dash": "solid"},
    "rgba(0,0,0,0)",
    "evenodd",
    "rect",
)

lines = []

for index, row in dfCoord.iterrows():

    if not index:
        continue

    if float(row["conf"]) < 50:
        continue

    if row["word_num"] > prev_row:
        if row["left"] - pre_row < 370:  # Mejorar este número
            line += row["text"] + " "
            min_x = min(min_x, row["left"])
            min_y = min(min_y, row["top"])
            max_x = max(max_x, row["left"] + row["width"])
            max_y = max(max_y, row["top"] + row["height"])
        else:
            lines.append((*preconfig, min_x, min_y, max_x, max_y, line[:-1] +␣
↪"\n"))
            line = row["text"] + " "
            min_x, min_y = row["left"], row["top"]
            max_x, max_y = row["left"] + row["width"], row["top"] + row["height"]
    else:
        lines.append((*preconfig, min_x, min_y, max_x, max_y, line[:-1] + "\n"))
        line = row["text"] + " "
        min_x, min_y = row["left"], row["top"]
        max_x, max_y = row["left"] + row["width"], row["top"] + row["height"]

    prev_row = row["word_num"]
    pre_row = row["left"]
```

0.7 Convert the list of tuples to DataFrame

[10]:

```
df_out = pd.DataFrame(
    lines,
    columns=[
        "editable",
        "xref",
        "yref",
        "layer",
        "opacity",
        "line",
        "fillcolor",
        "fillrule",
        "type",
        "x0",
        "y0",
        "x1",
        "y1",
        "text",
    ],
)
```

0.8 Create de figure for Plotly visualization with the image

[11]:

```
fig = px.imshow(img)
```

0.9 Adding all the boxes for every word/phrase/sentence founded with OCR

[12]:

```
for index, row in df_out.iterrows():
    fig.add_shape(
        type=row["type"],
        xref=row["xref"],
        yref=row["yref"],
        x0=row["x0"],
        x1=row["x1"],
        y0=row["y0"],
        y1=row["y1"],
        line=row["line"],
    )
```

0.10 Adding the feature to insert manually a box

[13]:

```
fig.update_layout(
    dragmode="drawrect",
    newshape=dict(line=dict(color="red", width=1)),
)
fig.update_layout(margin={"l": 0, "r": 0, "t": 0, "b": 0})
```

0.11 Configure the Dash Application and adding the figure

[14]:

```
config = {
    "modeBarButtonsToAdd": [
        # "drawline",
        # "drawopenpath",
        # "drawclosedpath",
        # "drawcircle",
        "drawrect",
        "eraseshape",
    ]
}

# Build App
app = dash.Dash(__name__)
app.layout = html.Div(
    [
        html.H4("Draw a shape, then modify it"),
        dcc.Graph(
            id="fig-image",
            figure=fig,
            config=config,
            style={"width": "150vh", "height": "150vh", "border": "1px black␣
 ↪solid"},
        ),
        dcc.Markdown("Characteristics of shapes"),
        html.Pre(id="annotations-pre"),
    ]
)


@app.callback(
    Output("annotations-pre", "children"),
    # Output('canvaas-table', 'data'),
    Input("fig-image", "relayoutData"),
    prevent_initial_call=True,
)
def on_new_annotation(string):
    # for key in relayout_data:
    if "shapes" in string:
        print(string)
        data = string["shapes"]
        print(data)
        data = pd.DataFrame.from_dict(data)
        print(data)

        data2 = pd.DataFrame()
        ReadingSection = pd.DataFrame()
        for index, row in data.iterrows():
            y1 = int(row["y0"])
            y2 = int(row["y1"])
            x1 = int(row["x0"])
            x2 = int(row["x1"])
            ReadingSection = img[y1:y2, x1:x2]
            text = pytesseract.image_to_string(ReadingSection, config="--psm 6")
            dfReadingSection = pd.DataFrame(StringIO(text))
            data2 = data2.append(dfReadingSection)
            print(data2)
        data2 = data2.to_dict(orient="records")
        return json.dumps(data2, indent=2)
    return dash.no_update


if __name__ == "__main__":
    app.run_server(debug=True)
```

```
Dash is running on http://127.0.0.1:8050/

 * Serving Flask app '__main__' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production
deployment.
```

```
   Use a production WSGI server instead.
 * Debug mode: on
```

```
An exception has occurred, use %tb to see the full traceback.

SystemExit: 1
```

```
C:\Users\tw\anaconda3\envs\test02-py39\lib\site-
packages\IPython\core\interactiveshell.py:3405: UserWarning:

To exit: use 'exit', 'quit', or Ctrl-D.
```

[ ]: