Introduction

In this notebook, we use Amazon Textract and Google Vision to provide a quick way of extracting text/tables from an image of a page.

Intended use: The intended use of this notebook is to quickly prototype. You should expect to modify the code in this notebook to suit your usecase.

Preparation: At a minimum, set a working folder, and make sure to add your API keys for both Textract and Google Vision. To do so, please follow the steps outlined here: https://github.com/MikeJGiordano/OCR_History/blob/main/ReadMe.md

This notebook contains four parts:

1. Unmodified image OCR. This is intended to quickly detect text from a single image.
    a. There is then an option to run one or both OCR tools on a whole folder.

2. Image preprocessing. This routine helps you to quickly preprocess a single image (adjust contrast, split image, etc).
    a. If you are satisfied with the preprocessing routine, it will give you the option to preprocess a whole folder.

3. Image preprocessing with text extraction. This runs the image modification from part 2 into the text detection from part 1.

4. Image preprocessing with table extraction from Textract. This uses the image modification from part 2 to extract a table using Textract.

Program Setup

There are 5 steps, marked A-E.

A: Import packages

In [1]:
```
import io
import json
import os

# if you don't have these packages use any package manager to install
# you can install all packages at once using the provided requirements.txt file
import cv2
import boto3
from google.cloud import vision

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import tqdm as tq

from PIL import Image, ImageDraw
from textractor import Textractor
from textractor.visualizers.entitylist import EntityList
from textractor.data.constants import TextractFeatures, Direction, DirectionalFinderType

# note: the following py file, you'll have to download
import preprocess as pp
```

B: Please set your working directories here

In [2]:
```
# please set the path to the folder containing your images here
input_folder = "images/"
# please set the path to a desired output folder here
output_folder = "output/"
```

C: Please set your main input file here

In [3]:
```
# set the filename to your image here
filename = "1888_Page_161.png"
```

D: Please authenticate Google Cloud

For help with Google Cloud, see https://github.com/MikeJGiordano/OCR_History/blob/main/Setup_Google_Cloud.md

In [4]:
```
#Authenticate Google Cloud here:

os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = 'ServiceAccountToken.json'
client = vision.ImageAnnotatorClient()
```

E: Please authenticate Amazon Textract

For help with Amazon Textract, see https://github.com/MikeJGiordano/OCR_History/blob/main/Setup_AWS_Root.md

In [5]:
```
#Authenticate AWS Textract in the console/terminal
```

Part 1: Basic text extraction

In [6]:
```
# plot the image, save .json outputs
pp.process_content(filename,
                   input_folder,
                   output_folder,
                   show_image=True,
                   use_google_vision=False,
                   use_textract=False,
                   verbose=True)
```

```
Setting all parameters=True gives a basic visualization of the outputs of both Cloud Vision, defaulted as the first image, and Textract, the second image. The .txt and .json outputs for both Cloud Vis
ion and Textract are saved in the output_folder. By setting a parameter=False, you can skip that function. For example, if use_textract=False and use_google_vision=True, this will not send the image t
hrough Textract, but will send the image through Google Vision.
```

You can use the next cell to get text and JSON files for the entire input folder through Google Vision, Textract, or both.

In [ ]:
```
# Batch process all images in the input folder, save text and JSON outputs to the output folder

pp.batch_ocr(input_folder,
                  output_folder,
                  use_google_vision=False,
                  use_textract=False)
```

Part 2: Preprocess images

Often, it helps to preprocess an image. Common routines are:

1. Adjusting contrast or brightness
2. Converting to grayscale
3. Cropping
4. Erasing margins
5. Splitting images

We now provide two examples:

1. Applying points 1-4
2. Preprocessing and splitting the image

Example 1: Full image

In [ ]:
```
# set the filename to your image here
filename = "1888_Page_161.png"
```

In [ ]:
```
#The next cell will apply the default preprocess settings to your image.
#If you are unsatisfied with those settings, it will provide instructions on how to make changes.
```

In [ ]:
```
#Preprocess a single image.
pp.preprocess_image(filename,
                    input_folder,
                    output_folder,
                    **pp.default);
```

Example 2: Split image

In [ ]:
```
# set the filename to your split image here
split_filename = "126.png"
```

In [ ]:
```
#The next cell will apply the default preprocess settings to your image.
#If you are unsatisfied with those settings, it will provide instructions on how to make changes.

pp.default['left_margin_percent'] = 30
pp.default['top_margin_percent'] = 5
```

In [ ]:
```
#Preprocess a split image.
pp.preprocess_image(split_filename,
                    input_folder,
                    output_folder,
                    **pp.default);
```

Part 3: Preprocessed Text Extraction

Example 1: Full image

In [ ]:
```
# using the above processing, the folder of modified images is located at:

modified_images = "output/modified_images/"

# Modification alters the name of the file to be:

modified_filename = 'modified_' + filename
```

In [ ]:
```
# plot the image, save .json outputs
pp.process_content(modified_filename,
                   modified_images,
                   output_folder,
                   show_image = True,
                   use_google_vision=False,
                   use_textract=True,
                   verbose=True)
```

Example 2: Split image

In [ ]:
```
# Modification splits the file into two and renames them:

modified_1_split = 'modified_1_' + split_filename
modified_2_split = 'modified_2_' + split_filename
```

In [ ]:
```
# plot the images, save .json and .txt outputs
pp.process_content(modified_1_split,
                   modified_images,
                   output_folder,
                   show_image = True,
                   use_google_vision=True,
                   use_textract=False,
                   verbose=True)

pp.process_content(modified_2_split,
                   modified_images,
                   output_folder,
                   show_image = False,
                   use_google_vision=False,
                   use_textract=False,
                   verbose=False)
```

You can use the next cell to get text and JSON files for the entire folder of modified images through Google Vision, Textract, or both.

In [ ]:
```
# Batch process all images in the modified folder, save .json outputs to the output folder

pp.batch_ocr(modified_images,
                  output_folder,
                  use_google_vision=False,
                  use_textract=False)
```

Part 4: Textract Table Extraction

Setup

Initialize Textractor client, modify region if required

In [ ]:
```
extractor = Textractor(profile_name="default")
```

Please specify the image you want to extract a table from.

In [ ]:
```
# using the above processing, the folder of modified images is located at:

modified_images = "output/modified_images/"

# Modification alters the name of the file to be:

modified_filename = 'modified_' + filename
```

Extract the tables

In [ ]:
```
pp.extract_table(extractor,
                   modified_filename,
                   modified_images,
                   output_folder);
```