Approach and Process

Approach 1: Using Pytesseract and Open CV Packages

The first process involves Pytesseract package, which is a Python wrapper of Tesseract-OCR from Google. It also involves Open CV Package for image processing.

Process:

1. The document is converted to image either by Python or by other means.
2. The converted document is sent to the image processing python function.
3. The image processing python function will do the following:
    a. Localize the Text within boundary using Pytesseract package
    b. Extract the data using Open CV
    c. Print out the output

Approach 2: Using Open CV Python Package and Masking [Working Model]

This process involves creating a mask and getting the values of the unmasked area.

Process:

1. The document is converted to image either by Python or by other means.
2. The converted document is sent to the image processing python function.
3. The image processing function will do the following:
    a. Read the Image
    b. Convert the Image to HSV
    c. Create a Lower and Upper Limits for the HSV Mask
    d. Create a mask using **inRange** function
    e. Find the contours in the document
    f. Grab the contours using Imutils package
    g. Perform Bitwise And operation on Original Image using the Mask

Further area of improvement can be with respect to finding contours, as in my case, even though the mask was able to find the required contours, during the bitwise operation it failed to take the last 2 contours.

The way I am thinking to productionize it, if that's the case then in AWS the following can be done.

1. Assuming that the documents are being pushed to S3 Bucket, a Cloudwatch Rule can be triggered based on this which calls Step Functions.
2. The Step functions state machine will have two lambdas associated with that. One will do the conversion of PDF to Image and other will do the actual masking and localization tasks.

3. So the step function triggers the first lambda in its workflow, which will get the document from S3 bucket, convert to Image using PDF2Image Python package and push the image to the same bucket but in different folder.
4. Once the task is completed by first lambda, the second lambda starts which will get the image from the folder in S3 bucket and does the image processing. The final output is either stored in a new S3 bucket or stored in the same bucket in different folder.
5. This task can be scaled if required, by assigning concurrency to lambda and increasing lambda memory.

Another feature addition to this might be to create a Deep Learning model that will be fed these highlighted documents or masks and once the model is trained, it can be used to predict the new documents and assess its performance.