!pip install importlib

!pip install pdfminer3k

import sys

import importlib

importlib.reload(sys)

import re

from pdfminer.pdfparser import PDFParser,PDFDocument

from pdfminer.pdfinterp import PDFResourceManager, PDFPageInterpreter

from pdfminer.converter import PDFPageAggregator

from pdfminer.layout import LTTextBoxHorizontal,LAParams

from pdfminer.pdfinterp import PDFTextExtractionNotAllowed

#from pdfminer.pdfpage import PDFPage

from pdfminer.pdfdevice import PDFDevice

from urllib.request import urlopen

from google.colab import drive

drive.mount('/content/drive')

%cd /content/drive/'My Drive'/'Peng'

pdf\_path = 'DiagnosticandstatisticalmanualofmentaldisordersDSM-5.pdf'

newname1 = 'newtest.txt'

def extract\_layout\_by\_page(pdf\_path):

    laparams = LAParams() # pdfminer.layout.LAParams(line\_overlap=0.5, char\_margin=2.0, line\_margin=0.5, word\_margin=0.1, boxes\_flow=0.5, detect\_vertical=False, all\_texts=False)

    fp = open(pdf\_path, 'rb')  #

    parser = PDFParser(fp)   # Create a PDF parser object associated with the file object.

    document = PDFDocument(parser) # Create a PDF document object that stores the document structure.

    #Connect analyzer to document object

    parser.set\_document(document)

    document.set\_parser(parser)

    # Check if the document allows text extraction. If not, abort

    document.initialize()

    if not document.is\_extractable:

        raise PDFTextExtractionNotAllowed

    rsrcmgr = PDFResourceManager()  # Create a PDF resource manager object that stores shared resources

    #device = PDFDevice(rsrcmgr) #'PDFDevice' object has no attribute 'get\_result'

    device = PDFPageAggregator(rsrcmgr, laparams=laparams)  # Create a PDF page aggregator object

    interpreter = PDFPageInterpreter(rsrcmgr, device) # Create a PDF interpreter object

    layouts = []

    count = 0

    # Process each page contained in the document

    for page in document.get\_pages():

        if count < 100:

            interpreter.process\_page(page)

            layouts.append(device.get\_result())

            count = count + 1

        else:

            break

    return layouts

def pdf\_to\_txt(pdf\_path, newname1):

    page\_layouts = extract\_layout\_by\_page(pdf\_path)

    with open(newname1, 'w', encoding='utf-8') as f:

        for current\_page in page\_layouts:

            # Get text

            for x in current\_page:

                if hasattr(x, "get\_text"):

                    # result.append(x.get\_text())

                    results = x.get\_text()

                    # print(type(results))

                    if results == ' \n':

                        continue

                    # results=list(results)

                    # with open(newname1, 'a', encoding='utf-8') as f:

                    f.write(results)

                    # f.close()

    with open(newname1, 'r', encoding='UTF-8-sig') as f:

        content = f.readlines()

        print(content)

    pattern = re.compile(r'\s{1,2}\n')

    newcontent = []

    for i in content:

        if bool(re.search(pattern, i)) is False:

            j = i.replace('\n', '').replace(' \n', '')

            newcontent.append(j)

        else:

            newcontent.append(i)

    f.close()

    print(newcontent)

    with open(newname1, 'w', encoding='UTF-8') as f:

        for i in newcontent:

            f.write(i)

    f.close()

    return newname1

pdf\_to\_txt(pdf\_path, newname1)