# -\*- coding: utf-8 -\*-

from \_\_future\_\_ import absolute\_import

from \_\_future\_\_ import division, print\_function, unicode\_literals

from sumy.parsers.html import HtmlParser

from sumy.parsers.plaintext import PlaintextParser

from sumy.nlp.tokenizers import Tokenizer

from sumy.summarizers.lsa import LsaSummarizer as Lsa

from sumy.summarizers.luhn import LuhnSummarizer as Luhn

from sumy.summarizers.text\_rank import TextRankSummarizer as TextRank

from sumy.summarizers.lex\_rank import LexRankSummarizer as LexRank

from sumy.summarizers.sum\_basic import SumBasicSummarizer as SumBasic

from sumy.summarizers.kl import KLSummarizer as KLsum

from sumy.nlp.stemmers import Stemmer

from sumy.utils import get\_stop\_words

from cStringIO import StringIO

from pdfminer.pdfinterp import PDFResourceManager, PDFPageInterpreter

from pdfminer.converter import TextConverter

from pdfminer.layout import LAParams

from pdfminer.pdfpage import PDFPage

from pdfminer.pdfparser import PDFParser

from pdfminer.pdfdocument import PDFDocument

import PyPDF2

import os

import time

import shutil

import sys, traceback

import xlrd

# Helper class used to map pages numbers to bookmarks

class BookmarkToPageMap(PyPDF2.PdfFileReader):

def getDestinationPageNumbers(self):

def \_setup\_outline\_page\_ids(outline, \_result=None):

if \_result is None:

\_result = {}

for obj in outline:

if isinstance(obj, PyPDF2.pdf.Destination):

\_result[(id(obj), obj.title)] = obj.page.idnum

elif isinstance(obj, list):

\_setup\_outline\_page\_ids(obj, \_result)

return \_result

def \_setup\_page\_id\_to\_num(pages=None, \_result=None, \_num\_pages=None):

if \_result is None:

\_result = {}

if pages is None:

\_num\_pages = []

pages = self.trailer["/Root"].getObject()["/Pages"].getObject()

t = pages["/Type"]

if t == "/Pages":

for page in pages["/Kids"]:

\_result[page.idnum] = len(\_num\_pages)

\_setup\_page\_id\_to\_num(page.getObject(), \_result, \_num\_pages)

elif t == "/Page":

\_num\_pages.append(1)

return \_result

outline\_page\_ids = \_setup\_outline\_page\_ids(self.getOutlines())

page\_id\_to\_page\_numbers = \_setup\_page\_id\_to\_num()

result = {}

for (\_, title), page\_idnum in outline\_page\_ids.items():

result[title] = page\_id\_to\_page\_numbers.get(page\_idnum, '???')

return result

#Functon to convert pdf into text

def convert(fname, pages=None):

if not pages:

pagenums = set()

else:

pagenums = set(pages)

# Check : print ('converting......')

output = StringIO()

manager = PDFResourceManager()

converter = TextConverter(manager, output, laparams=LAParams())

interpreter = PDFPageInterpreter(manager, converter)

infile = file(fname, 'rb')

for page in PDFPage.get\_pages(infile, pagenums):

interpreter.process\_page(page)

infile.close()

converter.close()

text = output.getvalue()

output.close

return text

def option(x):

return {

'1' : 'Luhn',

'2' : 'Lsa',

'3' : 'LexRank',

'4' : 'TextRank',

'5' : 'SumBasic',

'6' : 'KLsum',

'0' : 'exit',

}[x]

########################## Main Program ########################

#Set parameters

LANGUAGE = "English"

SENTENCES\_COUNT = 30

sourcePDFFile = raw\_input("Enter Soruce file with path \n")

if os.path.exists(sourcePDFFile):

print('Found source PDF file')

PDF\_SummaryDir= raw\_input("Enter Output Directory path \n")

chooseAlgo = option(raw\_input("Select Algorithm \n press 1 and enter for Luhn. \n press 2 and enter for Lsa. \n press 3 and enter for LexRank. \n press 4 and enter for TextRank. \n press 5 and enter for SumBasic.\n press 6 and enter for KLsum.\n press 0 and enter to exit. \n"))

#Check if the directory PDF\_summary exists or not

if not os.path.exists(PDF\_SummaryDir):

os.makedirs(PDF\_SummaryDir)

#create directories for output files

outputPDFDir = os.path.dirname(PDF\_SummaryDir + '\pdf\pdf\_split\_files\\')

if not os.path.exists(outputPDFDir):

os.makedirs(PDF\_SummaryDir + '\pdf\pdf\_split\_files\\')

outputTXTDir = os.path.dirname(PDF\_SummaryDir + '\Text\_Files\\')

if not os.path.exists(outputTXTDir):

os.makedirs(PDF\_SummaryDir + '\Text\_Files\\')

outputSummaryDir = os.path.dirname(PDF\_SummaryDir + '\Summary\\')

if not os.path.exists(outputSummaryDir):

os.makedirs(PDF\_SummaryDir + '\Summary\\')

#Name prefix for split files

outputNamePrefix = 'Split\_Chapter\_'

timeSuffixSummary = str(time.strftime("%d-%m-%Y\_%H.%M.%S"))

targetPDFFile = 'temppdfsplitfile.pdf' # Temporary file

# Append backslash to output dir ofor pdf if necessary

if not outputPDFDir.endswith('\\'):

outputPDFDir = outputPDFDir + '\\'

# Append backslash to output dir for txt if necessary

if not outputTXTDir.endswith('\\'):

outputTXTDir = outputTXTDir + '\\'

# Append backslash to output dir ofor pdf if necessary

if not outputSummaryDir.endswith('\\'):

outputSummaryDir = outputSummaryDir + '\\'

#Check and Verify if PDF is ready for splitting

while not os.path.exists(sourcePDFFile):

print('Source PDF not found, sleeping...')

#Sleep

time.sleep(10)

if os.path.exists(sourcePDFFile):

#print('Found source PDF file')

#Copy file to local working directory

shutil.copy(sourcePDFFile, targetPDFFile)

#Process file

#Create object and Open File in Read Binary Mode

pdfFileObj2 = open(targetPDFFile, 'rb')

pdfReader = PyPDF2.PdfFileReader(pdfFileObj2)

pdfFileObj = BookmarkToPageMap(pdfFileObj2)

#Get total pages

numberOfPages = pdfReader.numPages

i = 0

newPageNum = 0

prevPageNum = 0

newPageName = ''

prevPageName = ''

for p,t in sorted([(v,k) for k,v in pdfFileObj.getDestinationPageNumbers().items()]):

template = '%-5s %s'

# To Check Page number and Title of the Chapter Uncomment the following lines

## print (template % ('Page', 'Title'))

## print (template % (p+1,t))

newPageNum = p + 1

newPageName = t

if prevPageNum == 0 and prevPageName == '':

# First Page

prevPageNum = newPageNum

prevPageName = newPageName

else:

# Next Page

pdfWriter = PyPDF2.PdfFileWriter()

page\_idx = 0

for i in range(prevPageNum, newPageNum):

pdfPage = pdfReader.getPage(i-1)

pdfWriter.insertPage(pdfPage, page\_idx)

# Check : print('Added page to PDF file: ' + prevPageName + ' - Page #: ' + str(i))

page\_idx+=1

# Creating names of split files

pdfFileName = outputNamePrefix + str(str(prevPageName).replace(':','\_')).replace('\*','\_') + '.pdf'

txtFileName = outputNamePrefix + str(str(prevPageName).replace(':','\_')).replace('\*','\_') + '.txt'

# Writing each chapter to the .pdf file

pdfOutputFile = open(outputPDFDir + pdfFileName, 'wb')

pdfWriter.write(pdfOutputFile)

pdfOutputFile.close()

# Check : print('Created PDF file: ' + outputPDFDir + pdfFileName)

# Calling convert function and writing each chapter to the .txt file

txtOutputFile = open(outputTXTDir + txtFileName, 'w')

txtOutputFile.write(convert(outputPDFDir + pdfFileName))

txtOutputFile.close()

# Check :print('Created TXT file: ' + outputTXTDir + txtFileName)

# for plain text files create Summary

parser = PlaintextParser.from\_file(outputTXTDir + txtFileName, Tokenizer(LANGUAGE))

stemmer = Stemmer(LANGUAGE)

# Using LsaSummarizer to create summary

## Select from different algorithms to create summary by using different algorithms

if chooseAlgo == 'Lsa' :

summarizer = Lsa(stemmer)

elif chooseAlgo == 'LexRank':

summarizer = LexRank(stemmer)

elif chooseAlgo == 'TextRank':

summarizer = TextRank(stemmer)

elif chooseAlgo == 'Luhn':

summarizer = Luhn(stemmer)

elif chooseAlgo == 'SumBasic':

summarizer = SumBasic(stemmer)

elif chooseAlgo == 'KLsum':

summarizer = KLsum(stemmer)

else :

print ( 'Wrong Algorithm selected.')

sys.exit(0)

summarizer.stop\_words = get\_stop\_words(LANGUAGE)

# Open file in append mode so that summary will be added at the bottom of file

summaryOutputFile = open(outputSummaryDir + chooseAlgo + '\_Summary\_File' + timeSuffixSummary + '.txt','a')

for sentence in summarizer(parser.document, SENTENCES\_COUNT):

# Check : print (sentence)

summaryOutputFile.write(str(sentence))

# To create Separation between Chapters

summaryOutputFile.write('\n\n'+ 'Title : '+str(t)+'\n'+'\t')

summaryOutputFile.close()

# Check : print('Created TXT file: ' + outputSummaryDir + 'SummaryFile.txt')

i = prevPageNum

prevPageNum = newPageNum

prevPageName = newPageName

# Split the last page

pdfWriter = PyPDF2.PdfFileWriter()

page\_idx = 0

for i in range(prevPageNum, numberOfPages + 1):

pdfPage = pdfReader.getPage(i-1)

pdfWriter.insertPage(pdfPage, page\_idx)

# Check : print('Added page to PDF file: ' + prevPageName + ' - Page #: ' + str(i))

page\_idx+=1

pdfFileObj2.close()

# Delete temp file

os.unlink(targetPDFFile)