import re

import sys

import easygui as g

import os

import jieba

from pypinyin import lazy\_pinyin, Style

style = Style.TONE3

import time

starttime = time.perf\_counter()

class KanjiToPinyin(object):

def \_\_init\_\_(self):

self.shengmu = None

self.yunmu = None

self.tone = None

self.character = None

self.pinyin = None

self.braille = str

self.yunmu\_table = ['a', 'o', 'e', 'i', 'u', 'v', 'ai', 'ei', 'ui', 'ao', 'ou', 'iu', 'ie', 'ue', 'er', 'an',

'en', 'in', 'un',

'vn', 'ang', 'eng', 'ing', 'ong', 'ia', 'ian', 'iang', 'iao', 'iong', 'uai', 'ua', 'ue',

'uan', 'uang', 'uo', 'van']

def get\_pinyin(self, character):

self.\_\_init\_\_()

my\_re = re.compile(r'[A-Za-z]', re.S) # 该处的正则表达式的过滤已经失效 四行 大小写字母和数字的处理

res = re.findall(my\_re, character)

my\_re2 = re.compile(r'[0-9]', re.S)

res2 = re.findall(my\_re2, character)

if len(res) or len(res2):

return

else:

pass

self.break\_flag = False

self.character = Character(character)

#self.pinyin = p.get\_pinyin(self.character.string, tone\_marks='numbers') # , tone\_marks='marks',

self.pinyin=''.join(lazy\_pinyin(self.character.string, style=style))

if self.pinyin[-1] not in ['1', '2', '3', '4']:

self.pinyin=self.pinyin+'5'

def divide\_pinyin(self):

# global pos

if self.pinyin is None:

return

else:

pass

for letter in self.pinyin:

if letter in self.yunmu\_table:

pos = self.pinyin.index(letter)

break

if pos != 0:

self.shengmu = self.pinyin[0: pos]

self.yunmu = self.pinyin[pos:-1]

self.tone = self.pinyin[-1]

else:

self.yunmu = self.pinyin[pos:-1]

self.tone = self.pinyin[-1]

def get\_modified(self):

if self.pinyin is None:

return

else:

pass

# 盲文声母当中没有 y w 拼音当中的声母表存在 y w

if self.shengmu == 'y':

self.shengmu = None

if self.yunmu[0] == 'u': # 声母是y的时候，当韵母的第一个字母是u时，u转写为v,并且删去声母y

self.yunmu = 'v' + self.yunmu[1:] # 解决

elif self.yunmu[0] == 'i': # 声母是y的时候，当韵母的第一个字母是i时，保持韵母不变，删去声母

pass

else:

self.yunmu = 'i' + self.yunmu # 声母是y的时候，当韵母的第一个字母不是u也不是i的情况下，需要在韵母前添加i,删去声母y

elif self.shengmu == 'w':

self.shengmu = None

if self.yunmu[0] == 'u': # 声母是w的时候，韵母是u时，保持韵母不变，删去声母w

pass

else:

self.yunmu = 'u' + self.yunmu # 声母是w的时候，韵母不是u，在韵母之前添加u，删去声母w

if self.yunmu in ['iou', 'uei', 'uen']: # 汉语拼音规则

self.yunmu = self.yunmu[0] + self.yunmu[2] # iou iu uei ui uen un

if self.yunmu[0] == 'u': # u在和声母 j、q、x、y相拼的时候要去掉两点 y的情况在上面已经处理过 这里处理 j q x

if self.shengmu in ['j', 'q', 'x']: #+ [0]

self.yunmu = 'v'+ self.yunmu[1:] #+ self.yunmu[1:]

else:

pass

def transfer(self, character):

self.get\_pinyin(character)

self.divide\_pinyin()

self.get\_modified()

return (self.shengmu, self.yunmu, self.tone)

class quyindiao(object):

def \_\_init\_\_(self):

self.shengmu = None

self.yunmu = None

self.tone = None

self.pinyin = None

#2.1声母为 f 的音节，省写阴平符号

def f\_delete(self):

if self.shengmu=="f" and self.tone=="1":

self.tone="5"

#2.2 声母为 p、m、t、n、h、q、ch、r、c 的音节，省写阳平符号；音节 tóu的声调符号不省写

def pmt\_delete(self):

if (self.shengmu in ['p','m','t','n','h','q','r','c','ch']) and self.tone=="2":

if self.yunmu=='ou' and self.shengmu=='t':

pass

else:

self.tone="5"

#2.3声母为 b、d、l、ɡ、k、j、x、zh、sh、z、s 的音节，省写去声符号；音节 lè、zì的声调符号不省写。

def bdl\_delete(self):

if self.shengmu in ['b','d','l','g','k','j','x','z','s','sh','zh']and self.tone=='4':

if self.shengmu=='l' and self.yunmu=='e':

pass

elif self.shengmu=='z' and self.yunmu=='i':

pass

else:

self.tone="5"

#2.4 韵母自成的音节，省写去声符号，第 2.5（音节 yì、èr、wò、yè、yòu的声调符号不省写。）和 2.6（音节ē、é、ě、è 的声调符号不省写。 ） 条规定的除外

def ymzc\_delete(self):

self.pinyin = self.yunmu+self.tone

if self.shengmu==None and self.tone=='4':

if self.yunmu in ['i','er','uo','ie','iu'] or self.pinyin in ['e1','e2','e3','e4']:

pass

else:

self.tone="5"

#2.5 音节 yī、ér、wǒ、yě、yǒu 的声调符号省写

def yierwo\_delete(self):

self.pinyin = self.yunmu+self.tone

if self.shengmu==None and self.pinyin in ['i1','er2','uo3','iu3','ie3']:

self.tone="5"

#2.6音节ō、ó、ǒ、ò 的声调符号省写；音节ē、é、ě、è 的声调符号不省写。

def oooo\_delete(self):

self.pinyin = self.yunmu+self.tone

if self.shengmu==None and self.pinyin in['o1','o2','o3','o4']:

self.tone="5"

def transfer1(self, combination=(None, None, None)):

self.\_\_init\_\_()

self.shengmu = combination[0]

self.yunmu = combination[1]

self.tone = combination[2]

self.f\_delete()

self.pmt\_delete()

self.bdl\_delete()

self.ymzc\_delete()

self.yierwo\_delete()

self.oooo\_delete()

return (self.shengmu, self.yunmu, self.tone)

def transfer2(self, combination=(None, None, None)):

self.\_\_init\_\_()

self.shengmu = combination[0]

self.yunmu = combination[1]

self.tone = combination[2]

#self.f\_delete()

self.pmt\_delete()

self.bdl\_delete()

#self.ymzc\_delete()

#self.yierwo\_delete()

#self.oooo\_delete()

return (self.shengmu, self.yunmu, self.tone)

def transfer3(self, combination=(None, None, None)):

self.\_\_init\_\_()

self.shengmu = combination[0]

self.yunmu = combination[1]

self.tone = combination[2]

self.f\_delete()

self.pmt\_delete()

self.bdl\_delete()

#self.ymzc\_delete()

#self.yierwo\_delete()

#self.oooo\_delete()

return (self.shengmu, self.yunmu, self.tone)

class Pinyin2Braille:

def \_\_init\_\_(self):

self.shengmu = None

self.shengmu\_symbol = None

self.yunmu = None

self.yunmu\_symbol = None

self.tone = None

self.tone\_symbol = None

self.braille = None

self.dots = None

self.shengmu\_trans\_table = [

('b', 'b'), ('p', 'p'), ('m', 'm'), ('f', 'f'), ('d', 'd'), ('t', 't'), ('n', 'n'), ('l', 'l'), ('g', 'g'),

('k', 'k'), ('h', 'h'), ('j', 'g'), ('q', 'k'), ('x', 'h'), ('zh', '/'), ('ch', 'q'), ('sh', ':'),

('r', 'j'), ('z', 'z'), ('c', 'c'), ('s', 's'), #('y', 'i'), ('w', 'u'),

]

self.yunmu\_trans\_table = [

('a', '9'), ('o', '5'), ('e', '5'), ('i', 'i'), ('u', 'u'), ('v', '+'), ('ai', '['), ('ei', '!'),

('ui', 'w'), ('ao', '6'), ('ou', '('), ('iu', '\\'), ('ie', 'e'), ('er', 'r'), ('an', 'v'),

('en', '0'), ('in', '<'), ('un', '3'), ('vn', '\_'), ('ang', '8'), ('eng', '#'), ('ing', '\*'),

('ong', '4'), ('ueng', '4'), ('ian', '%'), ('ia', '$'), ('iang', 'x'), ('iao', '>'), ('iong', '?'),

('uai', 'y'), ('ua', '='),

('uan', ']'), ('uang', '7'), ('uo', 'o'), ('van', '&'), ('ve', ')'),('ue', ')')

]

self.tone\_trans\_table = [('1', 'a'), ('2', '1'), ('3', "'"), ('4', '2'),('5', '')]

def handle(self):

shengmu\_table = []

for group in self.shengmu\_trans\_table:

shengmu\_table.append(group[0])

yunmu\_table = []

for group in self.yunmu\_trans\_table:

yunmu\_table.append(group[0])

tone\_table = []

for group in self.tone\_trans\_table:

tone\_table.append(group[0])

if self.shengmu is None and self.yunmu is None:

self.braille = ''

elif self.shengmu is None:

self.shengmu\_symbol = ''

self.yunmu\_symbol = self.yunmu\_trans\_table[yunmu\_table.index(self.yunmu)][1]

self.tone\_symbol = self.tone\_trans\_table[tone\_table.index(self.tone)][1]

self.braille = self.yunmu\_symbol + self.tone\_symbol

elif self.shengmu in shengmu\_table:

if self.yunmu is None:

self.shengmu\_symbol = self.shengmu\_trans\_table[shengmu\_table.index(self.shengmu)][1]

self.braille = self.shengmu\_symbol + self.tone\_symbol

else:

if (self.shengmu in ['zh', 'ch', 'sh', 'z', 'c', 's']) and (self.yunmu == 'i'):

self.shengmu\_symbol = self.shengmu\_trans\_table[shengmu\_table.index(self.shengmu)][1]

self.tone\_symbol = self.tone\_trans\_table[tone\_table.index(self.tone)][1]

self.braille = self.shengmu\_symbol + self.tone\_symbol

else:

self.shengmu\_symbol = self.shengmu\_trans\_table[shengmu\_table.index(self.shengmu)][1]

self.yunmu\_symbol = self.yunmu\_trans\_table[yunmu\_table.index(self.yunmu)][1]

self.tone\_symbol = self.tone\_trans\_table[tone\_table.index(self.tone)][1]

self.braille = self.shengmu\_symbol + self.yunmu\_symbol + self.tone\_symbol

def set\_dots(self):

self.dots = ''

for i in self.braille:

self.dots = self.dots + braille2dots(i)

def transfer(self, combination=(None, None, None)):

self.\_\_init\_\_()

self.shengmu = combination[0]

self.yunmu = combination[1]

self.tone = combination[2]

self.handle()

self.set\_dots()

# print('2self.braille: ' + self.braille)

# print('2self.dots: ' + self.dots)

return (self.braille, self.dots)

class DotsBraille(object):

def \_\_init\_\_(self):

self.dots\_braille\_dict = {

# 'a': '1', '1': '2', 'b': '12', "'": '3', 'k': '13', '2': '23', 'l': '123', '@': '4', 'c': '14', 'i': '24',

# 'f': '124', '/': '34', 'm': '134', 's': '234', 'p': '1234', '"': '5', 'e': '15', '3': '25', 'h': '125',

# '9': '35', 'o': '135', '6': '235', 'r': '1235', '^': '45', 'd': '145', 'j': '245', 'g': '1245', '>': '345',

# 'n': '1345', 't': '2345', 'q': '12345', ',': '6', '\*': '16', '5': '26', '<': '126', '-': '36', 'u': '136',

# '8': '236', 'v': '1236', '.': '46', '%': '146', '[': '246', '$': '1246', '+': '346', 'x': '1346',

# '!': '2346', '&': '12346', ';': '56', ':': '156', '4': '256', '\\': '1256', '0': '356', 'z': '1356',

# '7': '2356', '(': '12356', '\_': '456', '?': '1456', 'w': '2456', ']': '12456', '#': '3456', 'y': '13456',

# ')': '23456', '=': '123456'

'a': '100000', '1': '010000', 'b': '110000', "'": '001000', 'k': '101000', '2': '011000', 'l': '111000', '@': '000100', 'c': '100100', 'i': '010100',

'f': '110100', '/': '001100', 'm': '101100', 's': '011100', 'p': '111100', '"': '000010', 'e': '100010', '3': '010010', 'h': '110010',

'9': '001010', 'o': '101010', '6': '011010', 'r': '111010', '^': '000110', 'd': '100110', 'j': '010110', 'g': '110110', '>': '001110',

'n': '101110', 't': '011110', 'q': '111110', ',': '000001', '\*': '100001', '5': '010001', '<': '110001', '-': '001001', 'u': '101001',

'8': '011001', 'v': '111001', '.': '000101', '%': '100101', '[': '010101', '$': '110101', '+': '001101', 'x': '101101',

'!': '011101', '&': '111101', ';': '000011', ':': '100011', '4': '010011', '\\': '110011', '0': '001011', 'z': '101011',

'7': '011011', '(': '111011', '\_': '000111', '?': '100111', 'w': '010111', ']': '110111', '#': '001111', 'y': '101111',

')': '011111', '=': '111111'

}

self.dots = str

self.braille\_sign = str

def braille2dots(self, string):

self.braille\_sign = string

if self.braille\_sign in self.dots\_braille\_dict.keys():

self.dots = self.dots\_braille\_dict[self.braille\_sign]

# print('3self.dots: ' + self.dots)

return self.dots

class Character(object):

def \_\_init\_\_(self, character):

self.string = character

def braille2dots(braille=str):

d = DotsBraille()

# print('4d.braille2dots(braille): ' + d.braille2dots(braille))

return d.braille2dots(braille)

def toBraille(string=''):

punctuation\_dict = {

'。': '"2',

'，': '"',

'？': '"\'',

'！': ';1',

'：': '-',

'、': '@',

'；': ';',

'（': ';\'',

'）': ',2',

'【': ';2',

'】': '2;',

'“': '^',

'”': '^',

'’': '^^',

'‘': '^^',

'《': '"-',

'》': '-1', #'》': '-1',

'<': '"\'',

'>': '\'"',

'.': '"2',

',': '"',

'?': '"\'',

'!': ';1',

':': '-',

';': ';',

'--': '36',

'(': ';\'',

')': '\';',

'[': ';2',

']': '2;',

'"': '^',

'': '^^',

'·':'-',

'-':'-',

'/':'#/',

'%':'%',

'+':'+',

'℃':'1"0,c',

'①':';-#1-2',

'②':';-#2-2',

'③':';-#3-2',

'④':';-#4-2',

'⑤':';-#5-2',

'Ⅰ':',i',

'Ⅱ':',ii',

'Ⅲ':',iii',

'a': ';a', # 小写字母模块 补充 小写字母 大写字母 数字 注意格式 参考通用盲文最后两个附录

'b': ';b', # 已完成 全部校对

'c': ';c',

'd': ';d',

'e': ';e',

'f': ';f',

'g': ';g',

'h': ';h',

'i': ';i',

'j': ';j',

'k': ';k',

'l': ';l',

'm': ';m',

'n': ';n',

'o': ';o',

'p': ';p',

'q': ';q',

'r': ';r',

's': ';s',

't': ';t',

'u': ';u',

'v': ';v',

'w': ';w',

'x': ';x',

'y': ';y',

'z': ';z',

'A': ',a', # 大写字母模块 已完成 全部校对

'B': ',b',

'C': ',c',

'D': ',d',

'E': ',e',

'F': ',f',

'G': ',g',

'H': ',h',

'I': ',i',

'J': ',j',

'K': ',k',

'L': ',l',

'M': ',m',

'N': ',n',

'O': ',o',

'P': ',p',

'Q': ',q',

'R': ',r',

'S': ',s',

'T': ',t',

'U': ',u',

'V': ',v',

'W': ',w',

'X': ',x',

'Y': ',y',

'Z': ',z',

'1': '#a', # 数字模块 已完成 #是为了区别相同盲文符号的其他意义 比如字母

'2': '#b',

'3': '#c',

'4': '#d',

'5': '#e',

'6': '#f',

'7': '#g',

'8': '#h',

'9': '#i',

'0': '#j',

'哟': '\\a',

'唷': '\\a',

'喲': '\\a',

}

output\_string = ''

output\_dots = ''

k = KanjiToPinyin()

p = Pinyin2Braille()

q = quyindiao()

string=string+' '

start = time.time()

for i in range(len(string)-1):

character0= string[i-1]

character= string[i]

character2= string[i+1]

if character in punctuation\_dict.keys():

output\_string += punctuation\_dict[character]

t = punctuation\_dict[character]

for x in t:

output\_dots += braille2dots(x)

elif character >= u'\u4e00' and character <= u'\u9fa5': # 汉字字符跳转到该语句

shenmu,yunmu,tone=k.transfer(character)

if (character2 >= u'\u4e00' and character2 <= u'\u9fa5') and shenmu!=None:#当后一个字是汉字

pinyin=shenmu+yunmu

if (pinyin in ['zhi','chi','shi','ri','zi','ci','si']):#如果这个字是sheng母自成音节

if k.transfer(character2)[0]==None: #且后一个字是韵母自成音节

outstring,outdots=p.transfer([shenmu,yunmu,tone])

output\_string +=outstring

output\_dots += outdots

else:

outstring,outdots=p.transfer(q.transfer2([shenmu,yunmu,tone]))

output\_string +=outstring

output\_dots +=outdots

else:

outstring,outdots=p.transfer(q.transfer3([shenmu,yunmu,tone]))

output\_string +=outstring

output\_dots +=outdots

else:

outstring,outdots=p.transfer(q.transfer1([shenmu,yunmu,tone]))

output\_string += outstring

output\_dots += outdots

else:

if character=='…':

if character2=='…':

output\_dots += '000010000010000010'

output\_string +='"""'

elif character0==' ' and character2==' ':

output\_dots += '000010000010000010'

output\_string +='"""'

else:

pass

elif character=='—':

if character2=='—':

output\_string += ',-'

output\_dots += '000001001001'

elif character0==' ' and character2==' ':

output\_dots += '000001001001'

output\_string += ',-'

else:

pass

elif character == ' ': # 空格直接在这里处理

output\_string += character

output\_dots += '000000'

else:

print('该字符串为非汉语体系字符：' + character)

output\_string += ' '

output\_dots += '000000'

pass

end = time.time()

print("循环运行时间:%.2f秒"%(end-start))

'''character=''.join(lb[-1])

if character in punctuation\_dict.keys():

output\_string += punctuation\_dict[character]

t = punctuation\_dict[character]

for y in t:

output\_dots += braille2dots(y)

elif character=='…':

output\_string += character

output\_dots += '000010000010000010'

elif character2=='—':

output\_string += character

output\_dots += '000001001001'

elif character == ' ': # 空格直接在这里处理

output\_string += character

output\_dots += '000000'

elif character >= u'\u4e00' and character <= u'\u9fa5': # 汉字字符跳转到该语句

output\_string += p.transfer(q.transfer1(k.transfer(character)))[0]

output\_dots += p.transfer(q.transfer1(k.transfer(character)))[1]

else:

print('该字符串为非汉语体系字符：' + character)

output\_string += ' '

output\_dots += '000000'

pass'''

#print('盲文点序: ' + output\_dots[0:])

#print('盲文ASCII码: ' + output\_string)

return output\_string, output\_dots[0:]

def TongYong\_mainthread():

file\_path = g.fileopenbox(msg='起点汉盲翻译软件V1.0版-请选择待翻译的文本文件', default='\*.txt') # , title='选择文件'

# print('绝对路径:' + file\_path)

f\_path, file\_name = os.path.split(file\_path)

# print('当前路径:' + f\_path)

# print('文件名称:' + file\_name)

f\_name = file\_name[0:-4]

# print (f\_name)

with open(file\_path, encoding='utf-8') as f:

text = f.read()

text2 = text.replace("\n", "") # 去除源文本中的换行

# print(str(text2))

text3 = text2.replace(" ", "") # 去除原始空格

# print(str(text3))

# 加载自定义词典

jieba.load\_userdict("user\_dict.txt")

# 采用默认方式\_精确模式

words = jieba.cut(text3, HMM=True)

# 分词 关闭jieba本身自带的HMM新词发现功能 HMM的未登录词识别方法

res = " ".join(words) # 加入分词空格

text4 = res.replace("… …", "……")

text5 = text4.replace("— —", "——")

filename = (f\_path + f'\\{f\_name}-分词结果.txt')

f = open(filename, "w+", encoding='utf-8')

f.write(text5) # 写入TXT

f.close()

print('文本分词连写处理已完成！')

output = toBraille(text5)

# print(output)

# 写为盲文点序文件

f = open(f\_path + f'\\{f\_name}-通用盲文点序翻译结果.txt', "w+")

f.write(output[1]) # output

f.close()

# 写为ASCII文件

f = open(f\_path + f'\\{f\_name}-通用盲文ASCII码翻译结果.txt', "w")

f.write(output[0])

f.close()

g.msgbox(' 翻译结束！', title="起点汉盲翻译软件V1.0版-通用盲文翻译", ok\_button="结束",

image="fin\_pic.jpg")

print('通用盲文（全带调）翻译结果文件创建成功！')

return None

if \_\_name\_\_ == '\_\_main\_\_':

TongYong\_mainthread()

# toBraille('')

# print('Total cost time:')

# endtime = time.perf\_counter()

# print(endtime-starttime)