**内容：**

对https://www.youtube.com/channel/UCkWbqlDAyJh2n8DN5X6NZyg（1965-2019年各季度最流行的编程语言）分析，通过OCR、图像分析方法，**还原成按年的静态表格**，动态展示，并分析一下相关数据变化的原因。按年方法为取每年Q4季度的数据。

**实验环境：**

主机配置：Inter Core i7-7700HQ CPU @ 2.80Ghz RAM 8.00GB

运行环境：WIN10 开发环境，python3.72，anaconda+jupyter notebook

1. **从视频中提取有用帧，保存为图片**
   1. **简单的间隔提取**

**思路：**

由于视频中各季度持续时间相等，故思路为获得每个季度持续的时长，然后定时截取视频帧。

实验FFmpeg+opencv截取帧，因为通过P发现每季度持续约38帧，故隔38帧截取一帧。（帧率为29.72....）。

**代码：**

import ffmpeg

import numpy

import cv2

import sys

def read\_frame\_as\_jpeg(in\_file, frame\_num):

"""

指定帧数读取任意帧

"""

out, err = (

ffmpeg.input(in\_file)

.filter('select', 'gte(n,{})'.format(frame\_num))

.output('pipe:', vframes=1, format='image2', vcodec='mjpeg')

.run(capture\_stdout=True)

)

return out

def get\_video\_info(in\_file):

"""

获取视频基本信息

"""

try:

probe = ffmpeg.probe(in\_file)

video\_stream = next((stream for stream in probe['streams'] if stream['codec\_type'] == 'video'), None)

if video\_stream is None:

print('No video stream found', file=sys.stderr)

sys.exit(1)

return video\_stream

except ffmpeg.Error as err:

print(str(err.stderr, encoding='utf8'))

sys.exit(1)

def improve(image):

image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

# 最终转为二值图

# ret, image = cv2.threshold(image, 127, 200, cv2.THRESH\_BINARY)

image = cv2.adaptiveThreshold(image, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, cv2.THRESH\_BINARY, 11, 2)

return image

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

file\_path = 'video.mp4'

video\_info = get\_video\_info(file\_path)

total\_duration = video\_info['duration']

avg\_frame\_rate = video\_info['avg\_frame\_rate']

i = 0

gap = 0

gap\_frame = 38 # 每季度持续38帧

while gap < float(total\_duration)\*30:

if i == 0:

gap += 2\*gap\_frame + 1

else:

gap += 4\*gap\_frame

out = read\_frame\_as\_jpeg(file\_path, (gap/30)\*(30000/1001))

image\_array = numpy.asarray(bytearray(out), dtype="uint8")

if image\_array.any() and (1965 + i <= 2019):

image = cv2.imdecode(image\_array, cv2.IMREAD\_COLOR)

#image = improve(image)

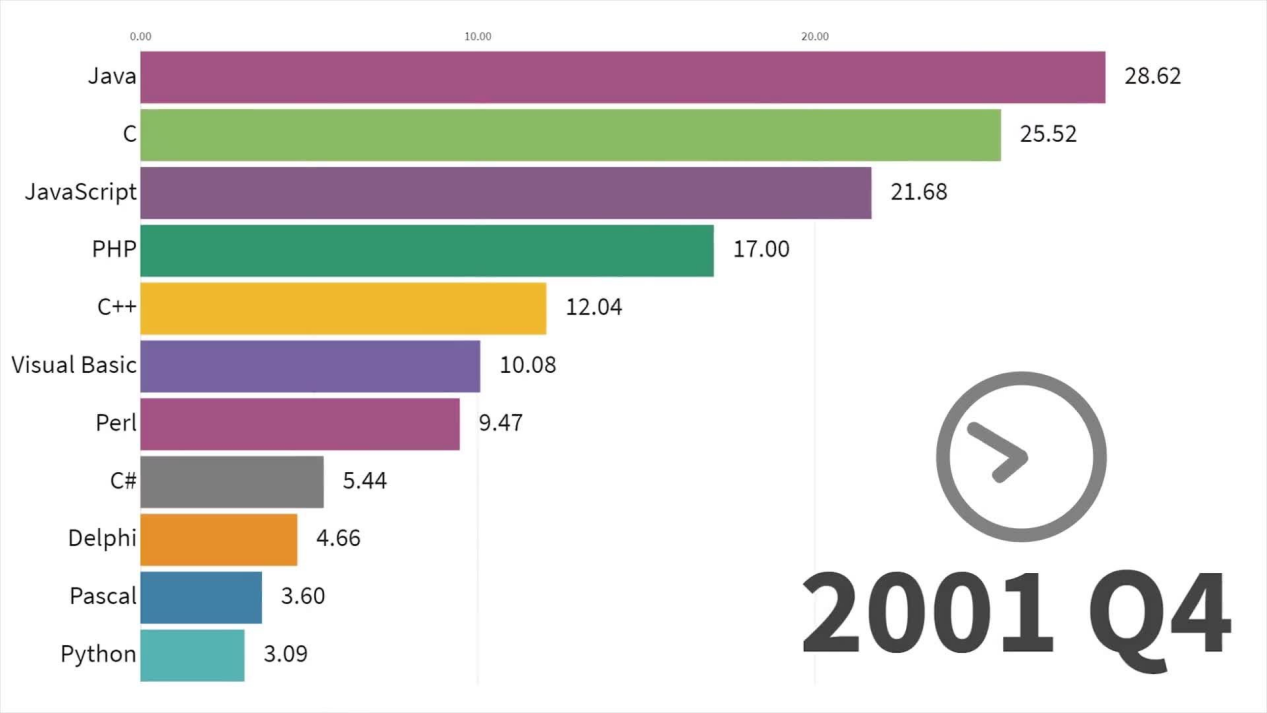
cv2.imwrite(str(1965 + i) + 'final Q4.jpg', image)

i += 1

运行结果：

输入文件：video.mp4（Popular Programming Languages 分辨率1080P）

输出文件：1965 Q4.jpg-2019 Q4.jpg



2001 Q4.jpg

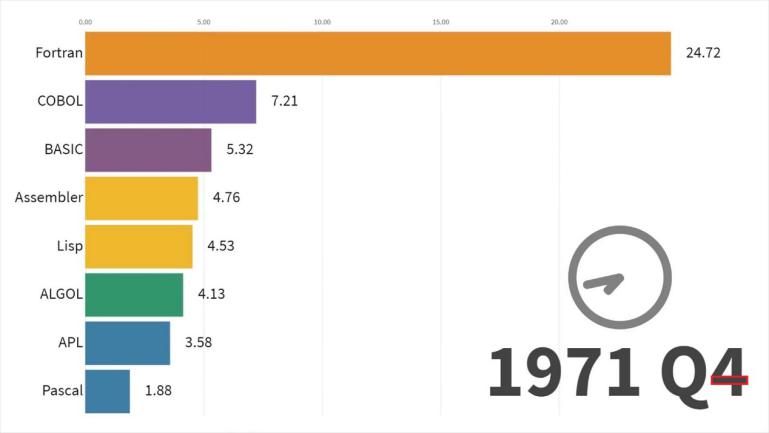
* 1. **实现程序自动判断并截取帧**

**思路：**

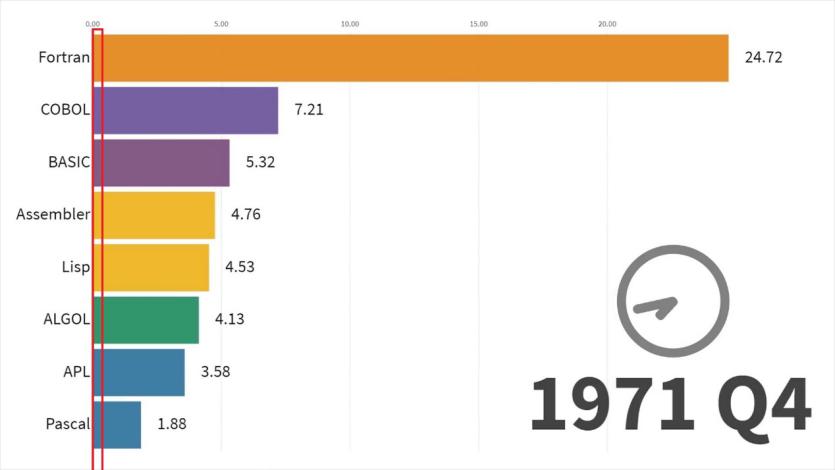
由于视频的分析最后是取每年的Q4季度的数据，所以只要通过控制截取帧的条件为该帧为Q4季度并且文字清晰可读（没有重叠）即可。

故做两方面判断：

1. 判断是否为Q4。因为右下角的时间信息位置固定，方法为取Q4的4的横线所在矩形（如图红色框），判断其中像素的颜色，>240时则认为是白色像素点，该矩形区白色像素点>400个则不是Q4，否则是Q4。



1. 判断是否清晰可读。清晰可读即不重叠，不重叠换一种思路就是图像中两柱子间空白区域比较短。



方法为取如图红色框所示的矩形区域（x范围218-229，y范围70-1080），自上而下分析每行，当该行白色像素点数量大于8则认为该行是白色的。这样自上而下分析，可以得到连续的白色行的长度，也就是矩形柱子间白色区域的大小。当每个白色区域在y方向长度都小于26时认为柱子无重叠或者重叠很少。

满足上述两个条件的帧可以保存。

在判断时为了加速，使用opencv提取帧，这样帧直接保存为numpy数组，可以直接判断，不用先保存。

为了提高运行速度，每隔5帧判断一次该帧是否符合条件。

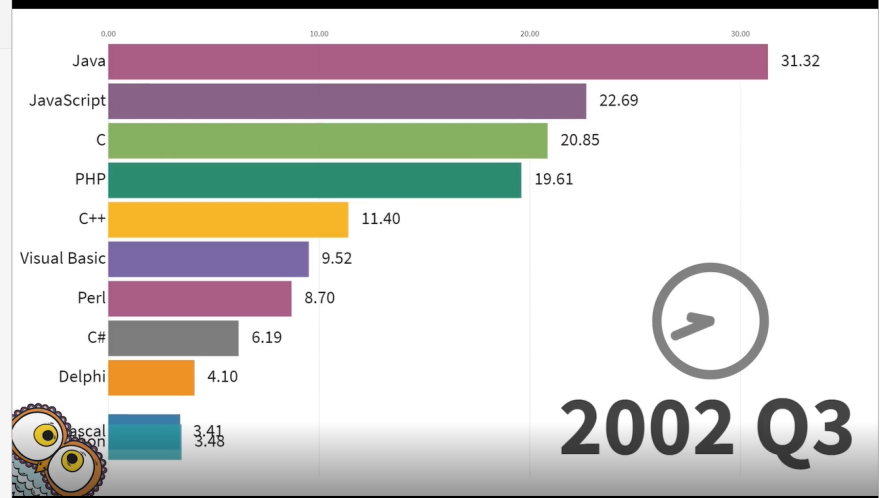
这样的问题是每年的Q4会截取很多张符合要求的图片，所以再增加一个判断，当当前帧符合条件且前一帧不符合条件时才截取。

另外由于2019年只有Q3季度，所以单独处理。

这样得到的直接是1965-2019，每年一张图片。

另注：

2003年视频中出现一只猫头鹰遮挡了文字但是经过上述判断得到的2003年Q4没有被遮挡，所以解决这个问题。



**代码：**

import numpy as np

import cv2

import sys

import os

import time

# 215,77 66 1776,939 1865,954

def judgeQ4(image): # 判断帧中内容是否为 Q4。判断标准为取4的-部分所在区域(1776,939 1865,954)，其中白色(>240)的像素点大于400则认为不是，否则认为是

image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

cnt = 0

for i in range(1776, 1865):

for j in range(939, 954):

if image[j][i] >= 240:

cnt += 1

if cnt >= 400:

return False

else:

return True

def judgecover(image): # 判断是否有覆盖。判断方法为截取高度在70-1080，x坐标在218-219的矩形部分，逐行判断，若该行白色像素点

# 数量>8则认为该行是白色的。取70开始是为了避免坐标轴上00,10,20会将上方的空白分成2部分

image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

(h, w) = image.shape # 然后计算出连续的白色行的长度，即为白色矩形的宽度，也就是数据的柱子之间白色的距离

h\_ = [0]\*h # [0,0,0,0,0,0,...] #当最大的间距<26时认为柱子无重叠或重叠的很少可以接受

for y in range(70, 1080):

cnt = 0

for x in range(218, 229):

if image[y, x] >= 240:

cnt += 1

if cnt >= 8:

h\_[y] = 1

break

len = 0

npp = []

for i in range(h):

if h\_[i] == 1:

len += 1

else:

if len != 0:

npp.append(len)

len = 0

if npp: # 可能是空的

npp = npp[0:-1]

# print(npp)

if npp:

if np.max(npp) >= 26:

return False

else:

return True

return True

return True

return getHProjection(image)

def getframes(path): # 每隔5帧截一帧。判断是否为Q4以及柱子的覆盖可以接收后保存

goal\_frame = np.array([])

cap = cv2.VideoCapture(path) # 返回一个cap对象，从0帧开始

if(cap.isOpened()):

print('c')

num = 0

i = 0

gap = 0

index = []

# gap\_frame = 38 # 每季度持续38帧

gap\_frame = 5 # 隔10帧提取一张图片

i = 0

# 假如gold\_img\_flag为0则说明前一张图片不符合条件

gold\_img\_flag = 0

# 必须要判断失败的图片差别较大的时候才进行保存,这样可以防止在排名发生变化的时候的较短间隔内截取两张图片

false\_gap = 0

while gap < 8480:

# print(i)

gap += gap\_frame

cap.set(cv2.CAP\_PROP\_POS\_FRAMES, gap) # 设置要获取的帧号

success, frame = cap.read() # read方法返回一个布尔值和一个视频帧。若帧读取成功，则返回True

# print(type(frame)) #numpy.ndarray

# print(frame.shape) #(1080,1920,3)

if judgeQ4(frame) and judgecover(frame):

false\_gap = 0

goal\_frame = frame

gold\_img\_flag = 1

# cv2.imwrite(str(i) + '.jpg', frame)

num += 1

index.append(i)

elif gold\_img\_flag == 1 and false\_gap > 5:

false\_gap = false\_gap + 1

print(1965+i)

cv2.imwrite('./judged/'+str(1965+i)+'.jpg', goal\_frame)

i = i+1

gold\_img\_flag = 0

else:

false\_gap = false\_gap + 1

return num, index

def get2019(): # 2019年没有Q4，特殊处理

gap = 8450

cap = cv2.VideoCapture(path) # 返回一个cap对象，从0帧开始

cap.set(cv2.CAP\_PROP\_POS\_FRAMES, gap)

success, frame = cap.read()

cv2.imwrite('./judged/'+str(2019) + '.jpg', frame)

def turn2years(index): # 上面得到的图片是按帧编号的，接下来取其中的55幅，代表各年。由于同一年的Q4帧间距离都比较近而不同年间距离比较远

# for i in range() #故在index中判断，较为接近的index认为是同一年的一组index，取同一组index中最大的一个index的图片作为代表该年的图片

# 取前后两个index差距大于20 (即帧差距大于100,帧率30000/1001，时间差约3.44秒)

savepath = './judged/'

last = -1

temp = 1965

for i in range(len(index)):

if index[i]-last > 20:

img = cv2.imread(str(last)+'.jpg')

cv2.imwrite(savepath+str(temp)+'.jpg', img)

temp += 1

last = index[i]

else:

last = index[i]

img = cv2.imread('2019.jpg')

cv2.imwrite(savepath+str(temp)+'.jpg', img)

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

start = time.time()

path = 'video.mp4'

# savepath='./judged/'+str(i)+'.jpg'

framenum, index = getframes(path)

get2019()

# index.append(2019)

# print(index)

# turn2years(index)

end = time.time()

print('totally cost', end-start) # 272.8170356750488 4分半

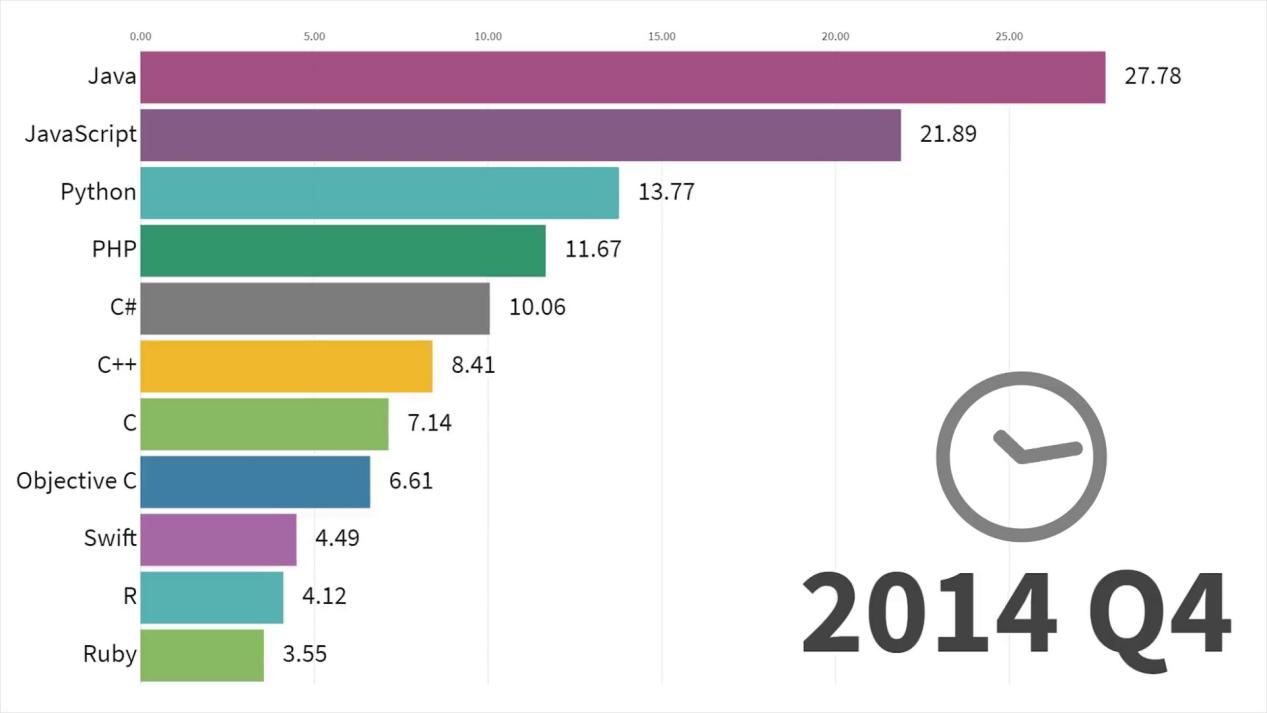
**运行结果：**

输入文件：video.mp4（Popular Programming Languages 分辨率1080P）

输出文件：1965.jpg-2019.jpg

执行时间：258.1404583454132s

（由于图片较多下面只展示一张）



2014.jpg

1. **使用OCR从提取的帧中识别文字，保存为CSV文件**
   1. **使用pytesseract-ocr识别**

该库之前是谷歌支持的库，但现在已经是个人维护了。

**思路：**

因为这个库识别率比较低，所以要先对图片进行去噪声处理。

去噪声步骤：

（1）对矩形柱子的去除

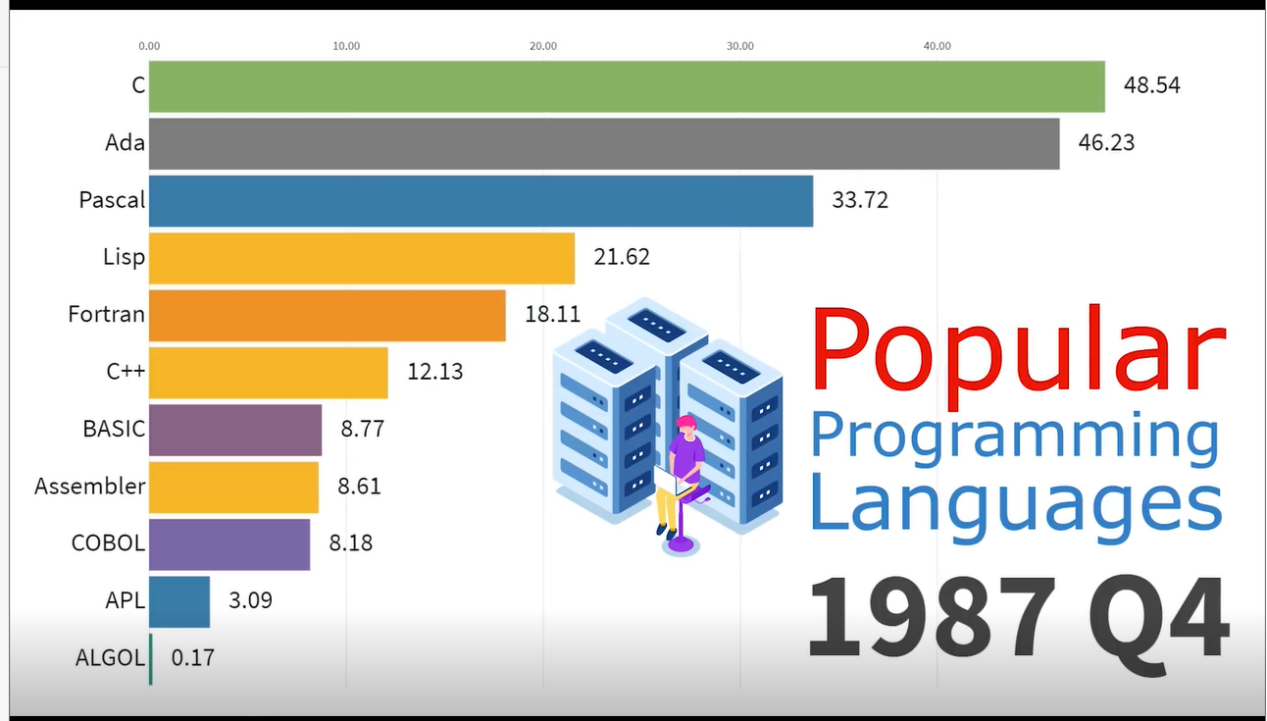
观察图片发现文字部分的像素值的三个通道之间至少有两个通道的差值<=1，而且图片中文字的颜色为浅灰色。使用上述两个特征对图片中的像素进行过滤，假如不满足条件的像素我们使用PIL库中的Image的putpixel方法对像素修改为白色（255，255，255）。

（2）对右下角和顶部的无用部分进行去除

先使用画图绘出mask图片，该图片与帧图片的大小相同，然后逐像素进行对比假如mask图片为白色的部分就将原图的对应位置改为白色，这种方式可以较为简单的去除一定区域内任意形状的图像。



由于之前的过滤条件设置不当导致有部分图片没有被正确过滤，所以使用了两个mask（如图），过滤的内容如下图中没有用的popular 等文字以及图片：



（3）将图片转化为二值图片并切割出单词

在去除无用部分后文字部分仍然会有一些噪点，因此我们将图片转化为二值图片并进行单词切割。

切割的方式是先按行进行切分，统计每行中白点的个数，做出白点关于行的分布图，最后利用这个特点进行横向切分，且分成几行。

然后对切割出的内容进行列的切分。方法类似对行进行切分，但是直接这样切分会导致每个单独的字符都被分别切割出来，所以应该将图片切割为单个单词。

每行只有两个单词，因此只用判断前一个字符的结束位置与后一个字符的开始位置的差距就可以判断出中间无字部分的开始，前后剩下的两个部分也就是那一行中的两个词。通过实验发现差距为20的时候认为间距差距较大，为单词间距离。

去噪声后调用OCR库进行识别。

Tesseract库的识别有一些选项，比如按照单个字符识别，按照一个单词识别等等。

设置识别模式为word。

同时该库可以添加白名单（也就是只识别其中的内容）。由于编程语言是数字字母和#等少量字符组成，而后面的数据为小数，只含0-9和 . ，所以设置白名单为这些内容。

经过上述去噪声和添加参数的处理，还会有一些问题。比如会将空格识别为换行符,以及其他不在白名单中的字符。tesseract4.0版本不支持whitelist，于是尝试了3.0版本与5.0版本的tesseract，但效果并不好，不如不加参数识别的效果。

另外还会出现把小数点识别为逗号的情况，把数字识别成字符的情况（比如零识别为o）

另外也尝试过直接对整张，处理过（去噪声步骤1，2）的图片进行识别但是效果不如切割过的图片更好。

**代码：**

**去除矩形柱子**

# coding = utf8

from PIL import Image

import os

import time

def masked\_image(image\_name, mask):

image = Image.open(image\_name)

size = image.size

width = size[0]

height = size[1]

for i in range(0, width):

for j in range(0, height):

data = image.getpixel((i, j))

mask\_data = mask.getpixel((i, j))

if mask\_data[0] == 255:

image.putpixel((i, j), (255, 255, 255, 0))

image.save(image\_name)

def to\_binary\_image(image\_name):

image = Image.open(image\_name)

size = image.size

width = size[0]

height = size[1]

for i in range(0, width):

for j in range(0, height):

data = image.getpixel((i, j))

# mask\_data = mask.getpixel((i, j))

sub1 = abs(data[0] - data[1])

sub2 = abs(data[1] - data[2])

sub3 = abs(data[2] - data[0])

if (sub1 <=1 or sub2 <=1 or sub3 <=1) and data[0]<110 and data[1]<110 and data[2] < 110:

image.putpixel((i, j), (0, 0, 0, 0))

# elif data[0] <= 110:

# image.putpixel((i, j), (0, 0, 0, 0))

else:

image.putpixel((i, j), (255, 255, 255, 0))

'''if data[0] >= 100 or data[1] >= 100 and data[2] >= 100:

image.putpixel((i, j), (255, 255, 255, 0))

else:

image.putpixel((i, j), (0, 0, 0, 0))'''

image.save(image\_name)

def convert(mask1, mask2):

for i in range(55):

image\_path = str(1965 + i) + '.jpg'

to\_binary\_image(image\_path)

if i <= 21 or i >= 35:

masked\_image(image\_path, mask2)

else:

masked\_image(image\_path, mask1)

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

start\_time = time.time()

mask1 = Image.open('mask1.jpg')

mask2 = Image.open('mask2.jpg')

convert(mask1, mask2)

end\_time = time.time()

print(end\_time-start\_time)

**转化为二值图并切割图片**

import cv2

import time

import numpy as np

def getHProjection(image):

'''水平投影'''

hProjection = np.zeros(image.shape, np.uint8)

# 图像高与宽

(h, w) = image.shape

# 长度与图像高度一致的数组

h\_ = [0]\*h

# 循环统计每一行白色像素的个数

for y in range(h):

for x in range(w):

if image[y, x] == 255:

h\_[y] += 1

# 绘制水平投影图像

return h\_

def getWProjection(image):

'''竖直投影'''

wProjection = np.zeros(image.shape, np.uint8)

# 图像高与宽

(h, w) = image.shape

# 长度与图像高度一致的数组

w\_ = [0]\*w

# 循环统计每一行白色像素的个数

for y in range(w):

for x in range(h):

if image[x, y] == 255:

w\_[y] += 1

# 绘制水平投影图像

return w\_

def character\_to\_word(W\_Start, W\_End):

"""

将w的切分变为以字为单位进行切分

"""

W\_word\_Start = []

W\_word\_End = []

for i in range(len(W\_Start)-1):

if W\_Start[i+1] - W\_End[i] > 20:

W\_word\_Start.append(W\_Start[0])

W\_word\_Start.append(W\_Start[i+1])

W\_word\_End.append(W\_End[i])

W\_word\_End.append(W\_End[-1])

break

return W\_word\_Start, W\_word\_End

def crop\_img(image\_name):

# 读入原始图像

# image\_name = '1965 Q4.jpg'

origineImage = cv2.imread(image\_name)

# 图像灰度化

#image = cv2.imread('test.jpg',0)

image = cv2.cvtColor(origineImage, cv2.COLOR\_BGR2GRAY)

# cv2.imshow('gray',image)

# 将图片二值化

retval, img = cv2.threshold(image, 127, 255, cv2.THRESH\_BINARY\_INV)

# cv2.imwrite('binary.jpg', img)

# 图像高与宽

(h, w) = img.shape

# 水平投影

H = getHProjection(img)

start = 0

H\_Start = []

H\_End = []

W\_Start = []

W\_End = []

# 根据水平投影获取垂直分割位置

for i in range(len(H)):

if H[i] > 0 and start == 0:

H\_Start.append(i)

start = 1

if H[i] <= 0 and start == 1:

H\_End.append(i)

start = 0

# 分割行，分割之后再进行列分割并保存分割位置

for i in range(len(H\_Start)):

# 设定阈值排除噪声

if H\_End[i] - H\_Start[i] < 5:

continue

start = 0

W\_Start = []

W\_End = []

# 获取行图像

cropImg = img[H\_Start[i]-3:H\_End[i]+3, :]

# cropImg\_name = image\_name + '\_' + str(i) + '\_row.jpg'

# print(cropImg\_name)

# cv2.imwrite(cropImg\_name, cropImg)

# 竖直投影

W = getWProjection(cropImg)

for j in range(len(W)):

if W[j] > 0 and start == 0:

W\_Start.append(j)

start = 1

if W[j] <= 0 and start == 1:

W\_End.append(j)

start = 0

# print('W\_Start:'+str(len(W\_Start)))

W\_word\_Start, W\_word\_End = character\_to\_word(W\_Start, W\_End)

for j in range(len(W\_word\_Start)):

crop\_w\_Img = cropImg[:, W\_word\_Start[j]-3:W\_word\_End[j]+3]

cropImg\_w\_name = image\_name + '\_' + \

str(i) + '\_row\_' + str(j) + '\_line.jpg'

# print(cropImg\_w\_name)

cv2.imwrite(cropImg\_w\_name, crop\_w\_Img)

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

start\_time = time.time()

for k in range(55):

image\_name = str(1965 + k) + '.jpg'

crop\_img(image\_name)

end\_time = time.time()

print(start\_time-end\_time)

**使用tesseract库识别分割的图片**

# -\*-encoding:utf-8-\*-

import pytesseract

from PIL import Image

import os

import time

def test():

dir\_path = './'

header = 0

now\_year = 1964

with open('result.txt', 'a+',encoding = 'utf-8') as f:

for i, image\_name in enumerate(sorted(os.listdir(dir\_path))):

# 假如没有表头要将表头写入

if header == 0:

f.write("year,language,percent,language,percent,language,percent,language,percent,language,percent,language,percent,language,percent,language,percent,language,percent,language,percent,language,percent")

header = 1

# 语言名和数字之前都要添加逗号

if 'line' in image\_name:

print(image\_name)

# 假如now\_year改变了就要写入年份数据

if str(now\_year) not in image\_name:

now\_year = now\_year + 1

f.write('\n')

f.write(str(now\_year))

f.write(',')

image = Image.open(image\_name)

if '\_0\_line' in image\_name:

text = pytesseract.image\_to\_string(image, lang='eng',\

config='--psm 10 --oem 3 -c tessedit\_char\_whitelist=ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz+#')

else:

text = pytesseract.image\_to\_string(image, lang='eng',\

config='--psm 10 --oem 3 -c tessedit\_char\_whitelist=0123456789.')

print(text)

f.write(str(text))

f.write('\n\n')

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

start\_time = time.time()

test()

end\_time = time.time()

print(end\_time - start\_time)

**运行结果：**

去除矩形条，对右下角和顶部的无用部分进行去除

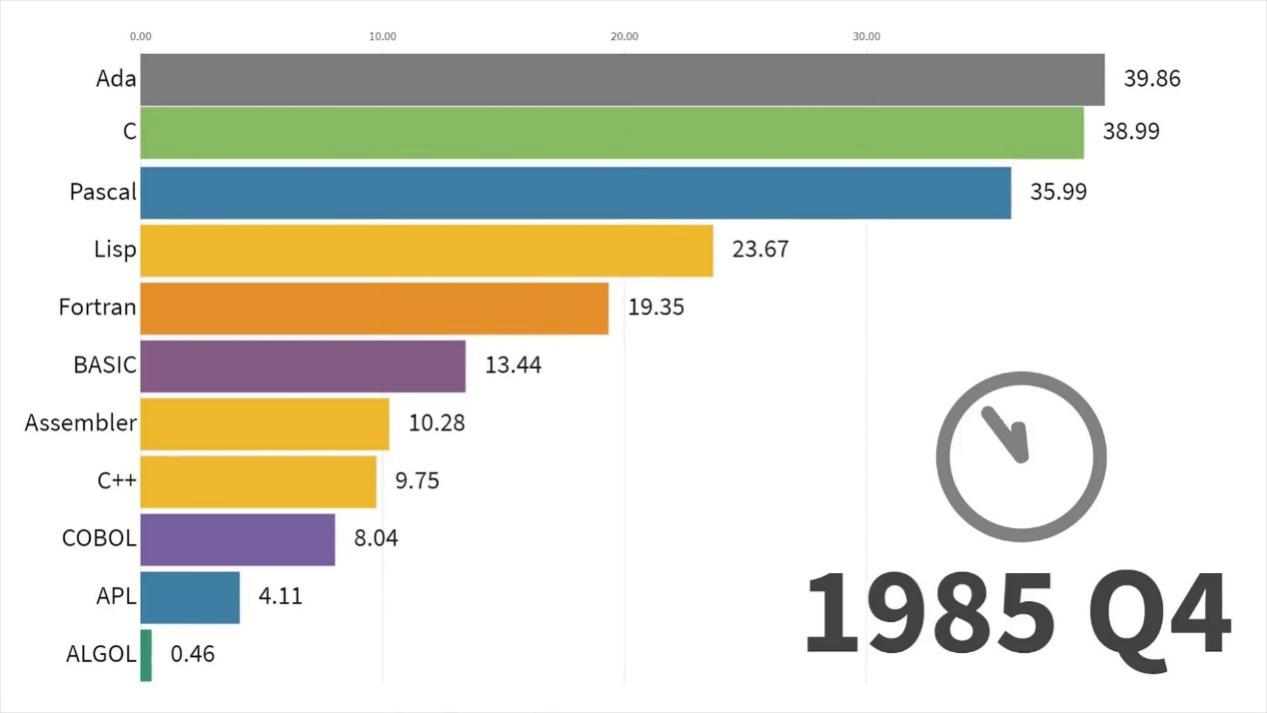
运行结果

主机配置：LENOVO RESCUER R720-15IKBN, CPU:INTER I5-7300HQ, RAM:8GB

运行环境：WIN10 开发环境：Anaconda python 3.73

执行时间：571.6861276626587s

输入文件：1965-2019.jpg，mask1，mask2.jpg



1985.jpg

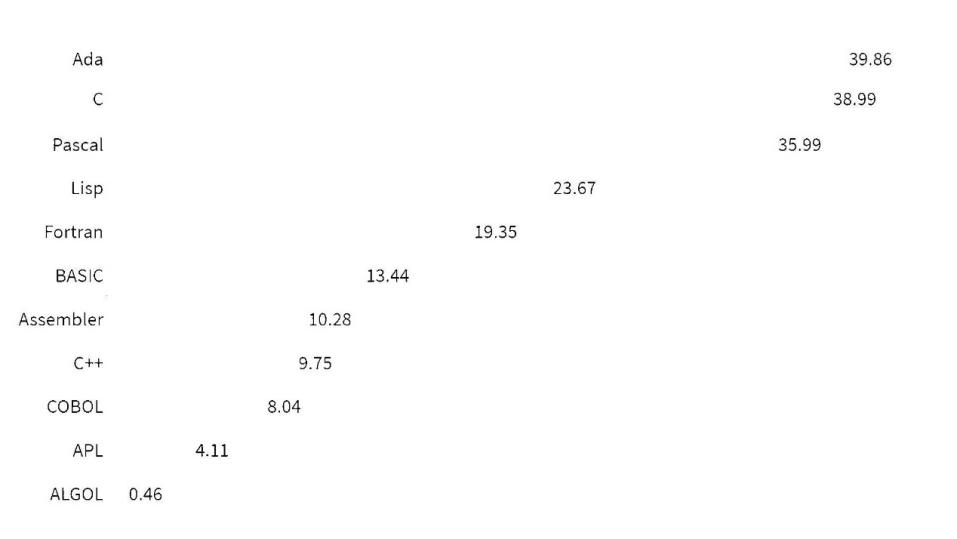


mask1.jpg



mask2.jpg

输出文件：



1985.jpg

过滤条件修改前：



过滤条件修改后：



*将图片转化为二值图片*

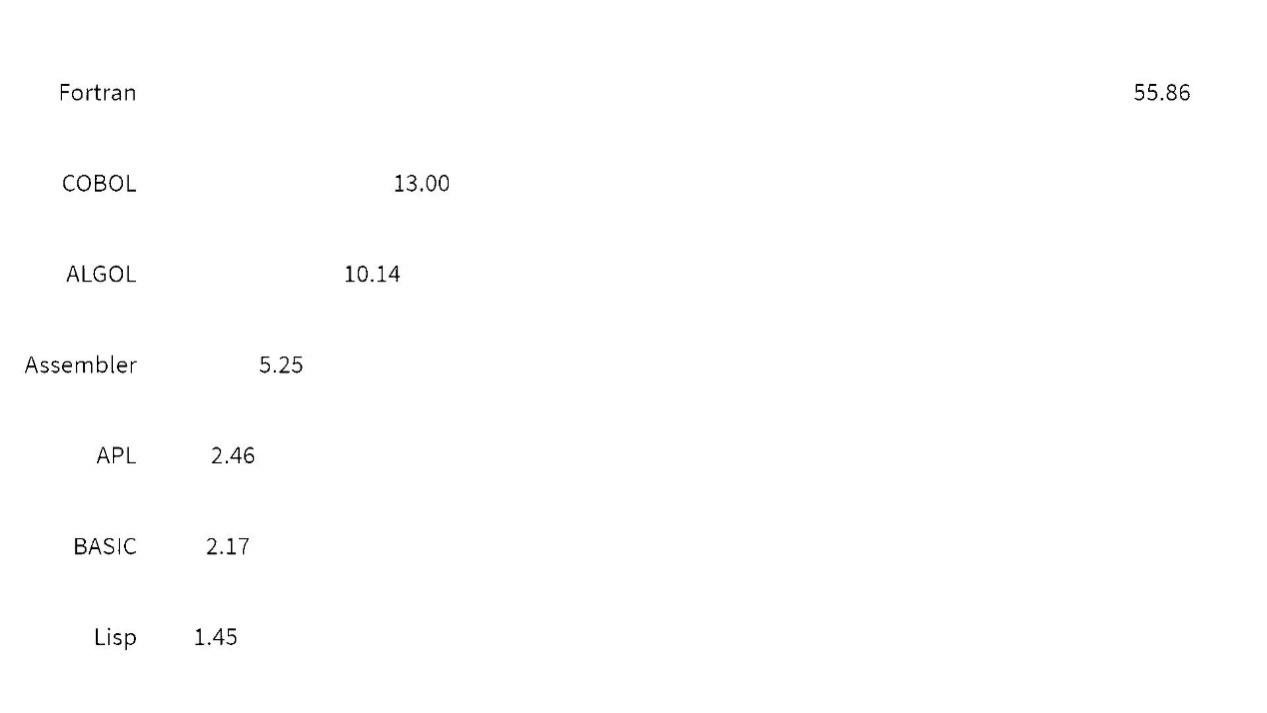
运行结果

主机配置：LENOVO RESCUER R720-15IKBN, CPU:INTER I5-7300HQ, RAM:8GB

运行环境：WIN10 开发环境：Anaconda python 3.73

执行时间：212.33676290512085s

输入图片：

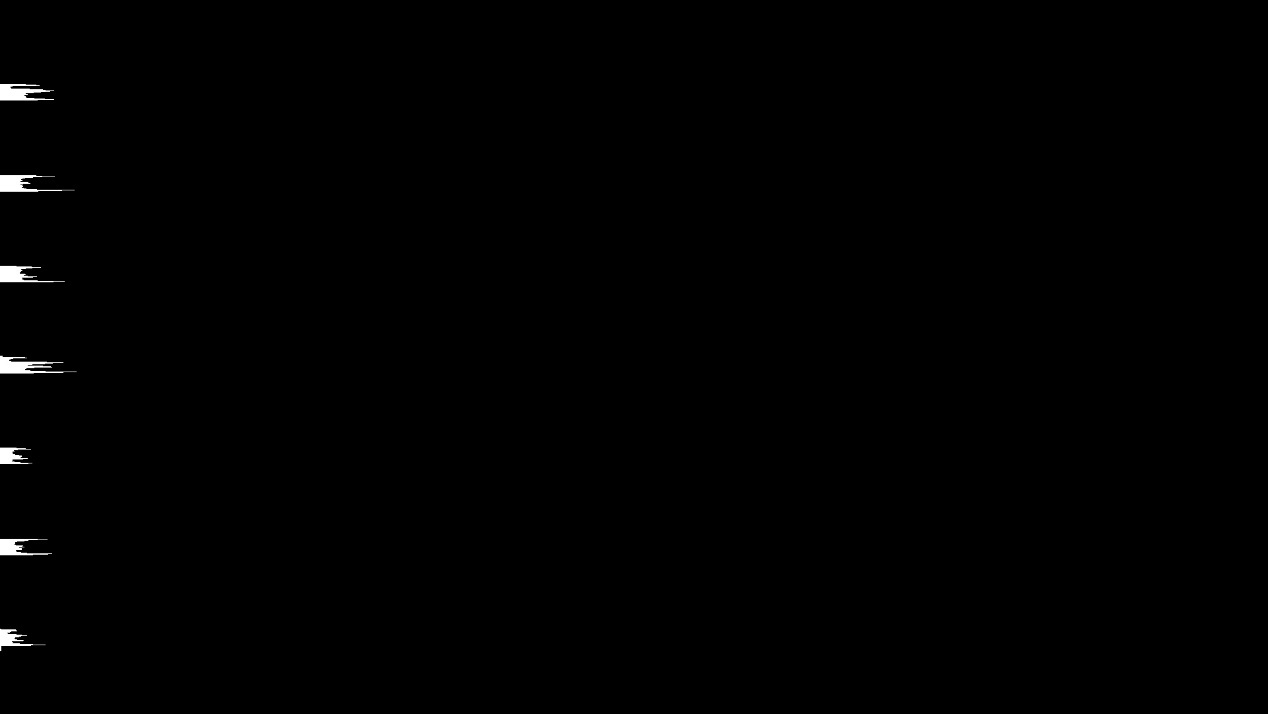


1965 Q4.jpg

输出图片：



binary.jpg（由输入图片转化成的二值图片）



hProjection2.jpg（二值图片的白色块的水平投影）

切割出的图片例子：



1965 Q4.jpg\_1\_row.jpg



wProjection\_1965 Q4.jpg\_1\_row.jpg



1965 Q4.jpg\_1\_row\_0\_line.jpg



1965 Q4.jpg\_1\_row\_1\_line.jpg

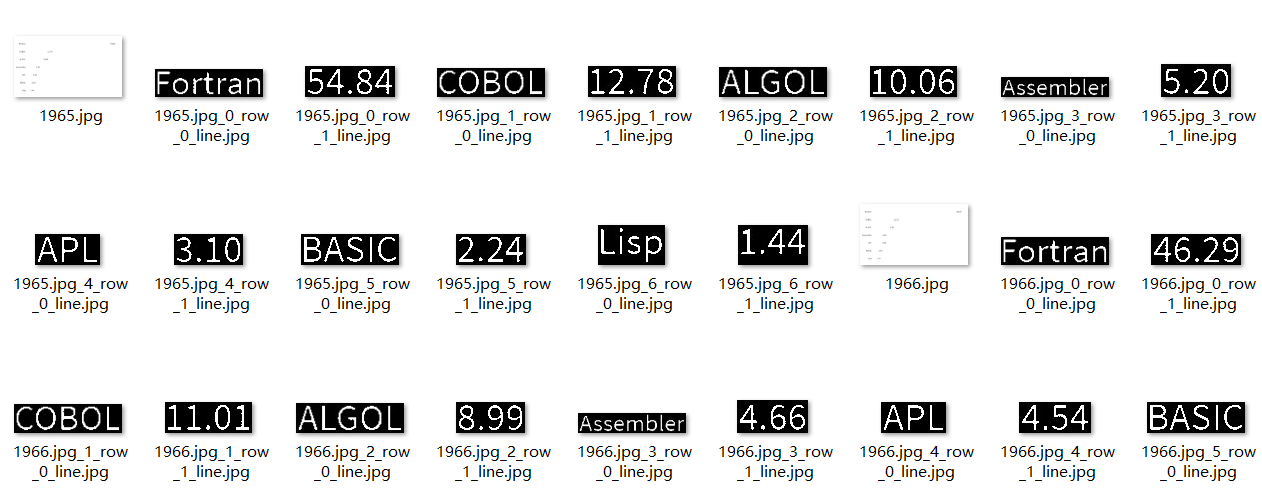
**调用OCR库进行识别**

主机配置：Inter Core i7-7700HQ CPU @ 2.80Ghz RAM 8.00GB

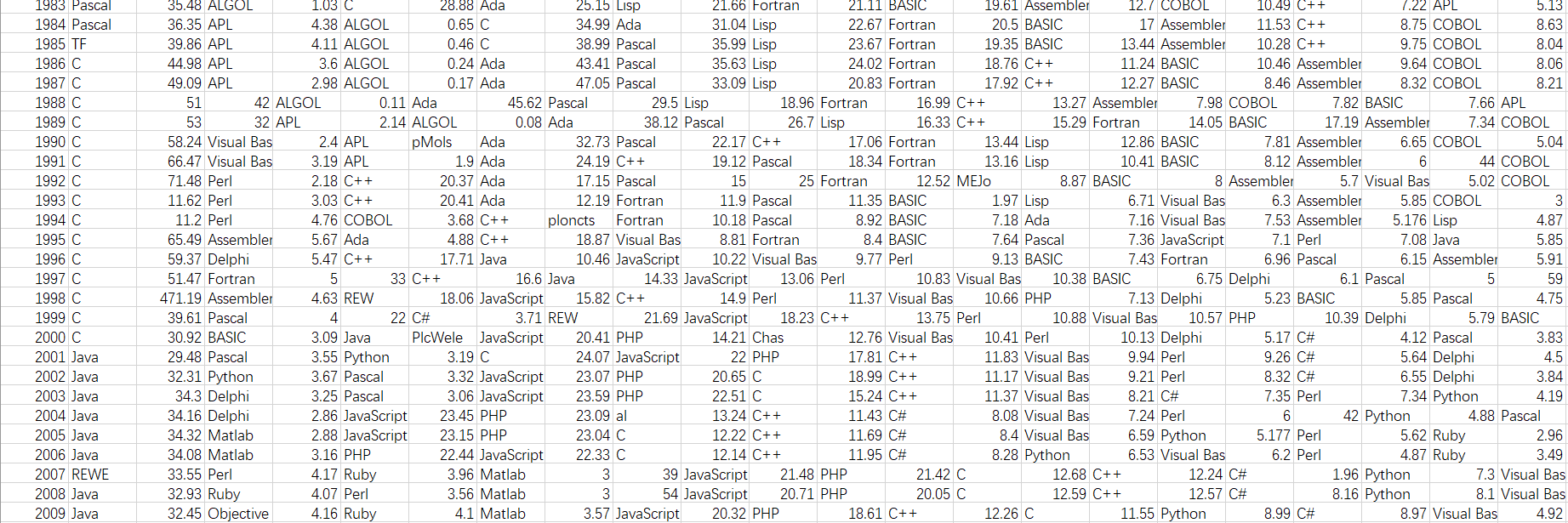
运行环境：WIN10 开发环境，python3.72

运行时间：769.1569538116455s

输入文件：切割好的图片

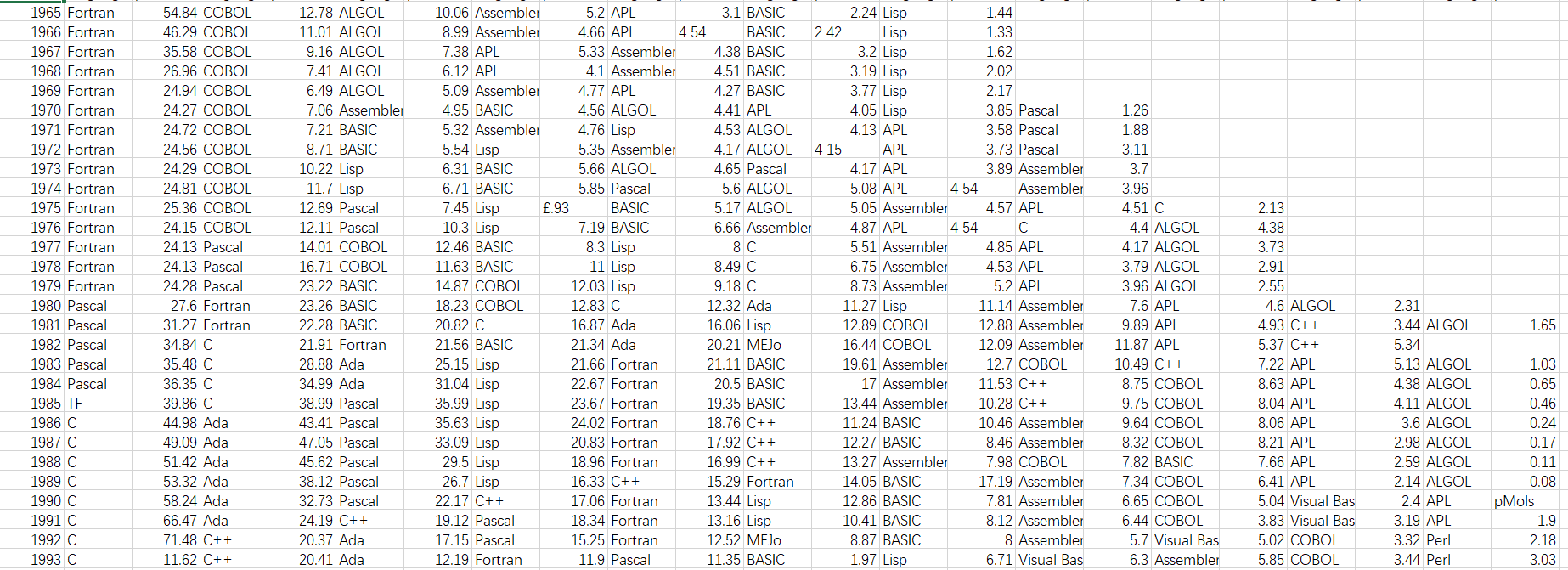


输出文件：result.csv（处理前）



处理指的是人工清洗

result.csv(处理后)



* 1. **使用百度api识别**

**思路：**

由于上述的tesseract的识别效果不佳，故尝试采用百度的ocr的api。尝试后发现百度api的basicAccurate识别模式（高精度模式）对彩色原图识别效果较好（优于灰度图，二值图），故直接将第一部中截取的1965-2019.jpg逐逐个调用api识别。

百度api返回的是类似字典（dict）的结构，其中的words项即为识别出的内容，提取写入文档即可。

由于百度api一次返回一张图的dict，故每次识别一张图就以追加方式写入文档。

这里先写入了txt文档，每个数据间用逗号隔开，这样可以方便的转化为csv（直接修改后缀）。

写入txt时先写文件头

"year,language,percent,language,percent,language,percent,language,percent,language,percent,language,percent,language,percent,language,percent,language,percent,language,percent,language,percent"

然后根据时间写入年份，然后再写入每张图片识别出的文字。

这里百度api识别也有一些问题.比如对单个字符识别不好（C,R语言）。C,R有的时候会丢失，C有时候会识别成E。由于只有这两种情况，所以编程就可以简单的进行处理，见代码的注释。同时对图片中的年份，mostpopular 等无用的文字进行过滤即可。

**代码：**

# -\*-encoding:utf-8-\*-

import pytesseract

from PIL import Image

from aip import AipOcr

# https://segmentfault.com/a/1190000015144721 general\_basic general accurate\_basic accurate

# 2017 {'words': 'C #'}, {'words': '8 .75'}, {'words': 'PHP '}, {'words': '8 .18'}, 有一次可能是网络问题？丢掉了8.75

config = {

'appId': '18058315',

'apiKey': 'h09BIebxL2CdOx6eGnDZpLb8',

'secretKey': 'qd6zu3NL2L04f0vToxO3RY5hCi49FtvQ'

}

client = AipOcr(\*\*config)

def get\_file\_content(file):

with open(file, 'rb') as fp:

return fp.read()

def main():

header=0

for k in range(55):

with open('testf.txt', 'a+') as f:

#first=1 # 第一个字符不需要写入 ','，否则要写 ', text'; csv以','分隔，有','会认为后面还有数据;还需要表头

if header==0:

f.write("year,language,percent,language,percent,language,percent,language,percent,language,percent,language,percent,language,percent,language,percent,language,percent,language,percent,language,percent")

f.write('\n')

header=1

#data=f.read()

#print(data)

image\_path=str(1965 + k) + '.jpg'

image = get\_file\_content(image\_path)

cnt=0 #丢失字符个数

last=-1 #last==0，上一个是数字；==1，上一个是字符

flag=-1 #一开始是连续的数字，需要跳过;

options={}

options["language\_type"]="ENG"

#options["recognize\_granularity"]="big"

result = client.basicAccurate(image,options) #高精度会把数字分开。如2000-> 2 000

#result = client.basicGeneral(image,options) #有时候会把C++识别为 +t enhancedGeneral，...

if result.get('words\_result')!=None: #可以直接写入CSV，这里先写到TXT

print(result['words\_result'])

tpstr=str(1965+k)

yearnum=1965+k

if yearnum <2000:

yearnum='1 '+str(yearnum%1000)

elif yearnum <2010:

yearnum='2 00'+str(yearnum%2000)

else:

yearnum='2 0'+str(yearnum%2000)

print(yearnum)

f.write(tpstr)

for i in result.get('words\_result'):

tpstr=i.get('words')#str

#print(tpstr+"x") #跳过最后的年份。如2009 xxx；以及populao languages

if tpstr.find(yearnum)!=-1 or tpstr.find('Popular')!=-1 or tpstr.find('Programming')!=-1 or \

tpstr.find('Languages') != -1:

continue

if flag==-1: #跳过开始的尺度 0.00 10.00 20.00 ...

if tpstr.find(".00")!=-1:

print("???")

continue

else:

flag=1

if tpstr=='C ' or tpstr=='E ': #有时候会把单独的字符C认为是E.单个字符C,R有时会丢失;C排第一时可能会掉

cnt+=1

f.write(",C ")

last=1

elif tpstr=='R ':

cnt+=1

f.write(","+tpstr)

last=1

elif tpstr.find('.')!=-1 and (last==-1 or last==0): #tpstr.find('.')!=-1 识别出了数字且上一个也是数字

if cnt ==0:

f.write(",C ")

tpstr=tpstr[0]+tpstr[2:] #识别结果为 类似 2 1.01，中间有空格

f.write(',')

f.write(tpstr)

cnt+=1

last=0

else:

f.write(",R ")

tpstr=tpstr[0]+tpstr[2:] #识别结果为 类似 2 1.01，中间有空格

f.write(',')

f.write(tpstr)

last=0

elif tpstr.find('.')!=-1:

tpstr=tpstr[0]+tpstr[2:] #识别结果为 类似 2 1.01，中间有空格.只能写str。后续再csv改为float

f.write(',')

f.write(tpstr)

last=0

else:

f.write(','+tpstr)

last=1

print("finish\n\n")

f.write("\n")

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

start=time.time()

main()

end=time.time()

print('totally cost',end-start) #272.8170356750488 4分半

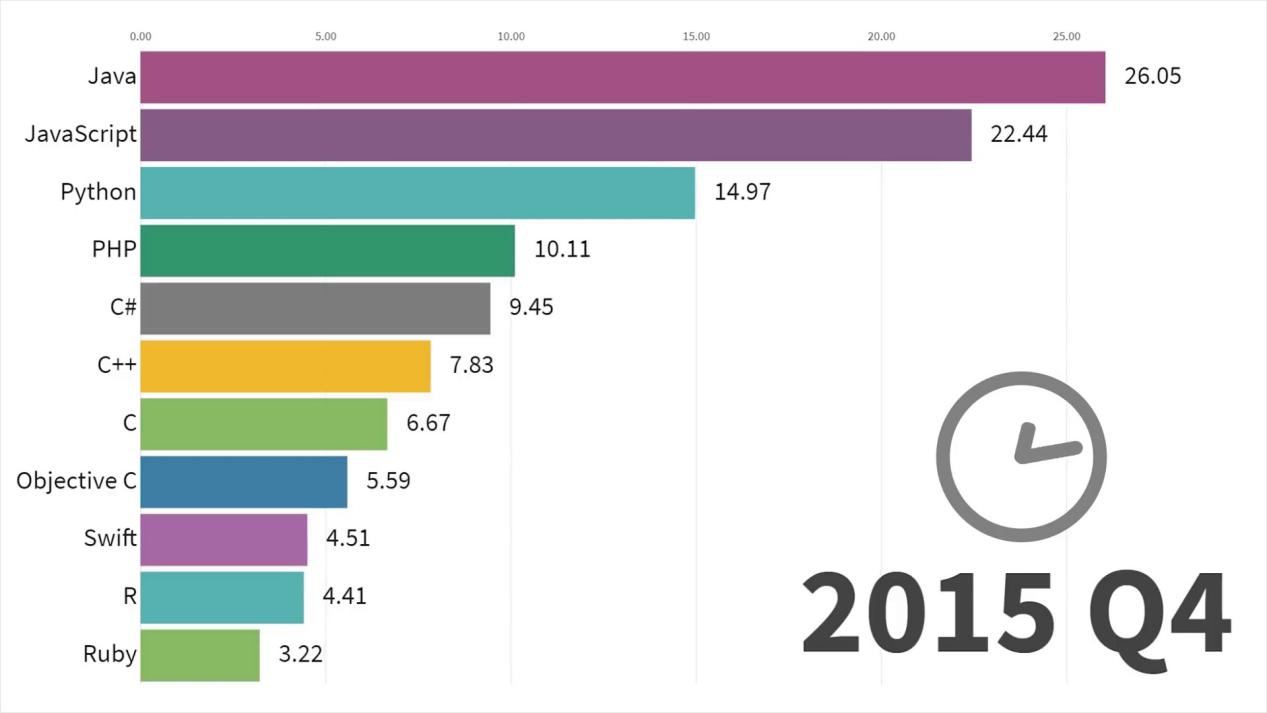
**运行结果：**

主机配置：Inter i5-7300HQ CPU @ 2.50GHz

运行环境：WIN10，anaconda3.73

运行时间：63s

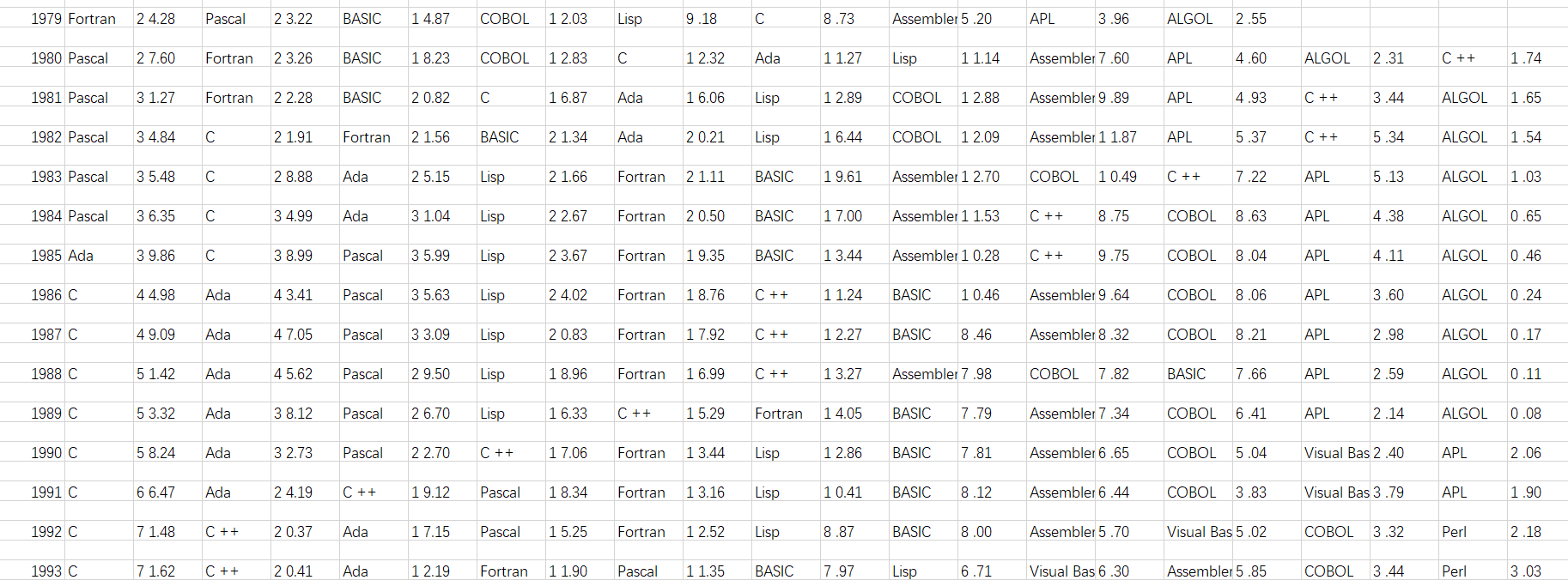
输入文件：由于文件较多这里选取其中一个作为示例

****

2015.jpg

**输出结果：**

testf.txt，修改后缀即可得到csv文件



1. **对分析结果进行动态展示**

**思路：**

由于百度api识别效果较好，故采用百度api识别得到的csv文件进行展示。

展示内容

1. 将第一部分自动截取的帧合并为视频展示
2. 将csv文件，按年提取，每年利用matplotlib绘制出柱状图，合并为视频
3. 根据输入，按列提取csv文件，并绘制出1965-2019每年第k流行的语言的折现变化图，并合并为视频

**代码：**

**画出每年的直方图和流行语言变化的折线图**

import os

import numpy as np

import pandas as pd

import csv

import matplotlib.pyplot as plt

from matplotlib import animation

import time

def anime(path): #顺序播放

plt.figure()

for i in range(55):

plt.subplot(1,1,1)

num=1965+i

data = pd.read\_csv(path) #列数不同有问题

data=data.loc[num-1965] #0指数据的第一行

lannum=int((data.count()-1)/2)

y1=[]

y1.append(data['percent'])

for i in range(lannum-1):

column='percent.'+str(i+1)

y1.append(data[column])

xr=[]

xr.append(data['language'])

for i in range(lannum-1):

column='language.'+str(i+1)

xr.append(data[column])

plt.bar(xr,y1,label=data['year'])

plt.xlabel('language')

plt.ylabel('percent')

plt.title(str(num)+'\'s popular language')

plt.xticks(rotation=-45)

plt.ion()

plt.pause(0.5)

plt.clf()

return

def specific(path,num): #特定一年的展示

data = pd.read\_csv(path) #列数不同有问题

data=data.loc[num-1965] #0指数据的第一行

#print(data)

#print(data.count())

lannum=int((data.count()-1)/2)

#print(data)

#print(data.dtypes)

y1=[]

y1.append(data['percent'])

for i in range(lannum-1):

column='percent.'+str(i+1)

y1.append(data[column])

xr=[]

xr.append(data['language'])

for i in range(lannum-1):

column='language.'+str(i+1)

xr.append(data[column])

plt.bar(xr,y1,label=data['year'])

plt.xlabel('language')

plt.ylabel('percent')

plt.title(str(num)+'\'s popular language')

plt.xticks(rotation=-45)

#plt.legend()

plt.show()

#plt.pie(x=y1, labels=xr)

#plt.show()

return

def mostpopular(path,num): #第num流行的语言

kth=num

data = pd.read\_csv(path) #列数不同有问题

#data=pd.read\_csv(path,header=line)

#data.describe()

#print(data)

#data.hist(bins=100,figsize=(15,10)) #bins表示直方图中柱子的数量，figsize是每张图的大小

#plt.show()

if kth==1:

column='language'

else:

column='language.'+str(kth-1)

x=data['year']

y1=data[column]

plt.plot(x,y1,label='most popular')

plt.xlabel('year')

plt.ylabel('language')

plt.title(str(kth)+'th popular language')

#plt.legend()

filename=str(num)+'th popular language'

plt.savefig(filename)

plt.show()

#my\_x\_ticks=np.arrange(1960,2020,5)

#plt.xticks(my\_x\_ticks)

#my\_yticks=np.arrange(-15000,10000,1000)

#plt.yticks(my\_yticks)

def txt2csv(path):

portion = os.path.splitext(path)#将文件名拆成名字和后缀

print(portion)

newname = portion[0] + ".csv"

os.rename(path, newname)#修改

data = pd.read\_csv(newname) #列数不同有问题

print(data.dtypes)

data[['percent','percent.1','percent.2','percent.3','percent.4','percent.5','percent.6',\

'percent.7','percent.8','percent.9','percent.10']] = \

data[['percent','percent.1','percent.2','percent.3','percent.4',\

'percent.5','percent.6','percent.7','percent.8','percent.9','percent.10']].astype(float)

print(data.dtypes)

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

path='testf.txt'

#path=txt2csv(path)

#anime(path)

#mostpopular(path,1)

#specific(path,2019)

path='testf.csv'

#anime(path)

specific(path,1965)

for i in range(5):

mostpopular(path,i+1)

**图片合并为视频:**

import cv2

import numpy as np

import os

img\_array = []

for i in range(1, 6):

filename = str(i) + 'th popular language.png'

img = cv2.imread(filename)

height, width, layers = img.shape

size = (width, height)

for \_ in range(20):

img\_array.append(img)

out = cv2.VideoWriter('line chart.avi', cv2.VideoWriter\_fourcc(\*'DIVX'), 15, size)

for i in range(len(img\_array)):

out.write(img\_array[i])

out.release()

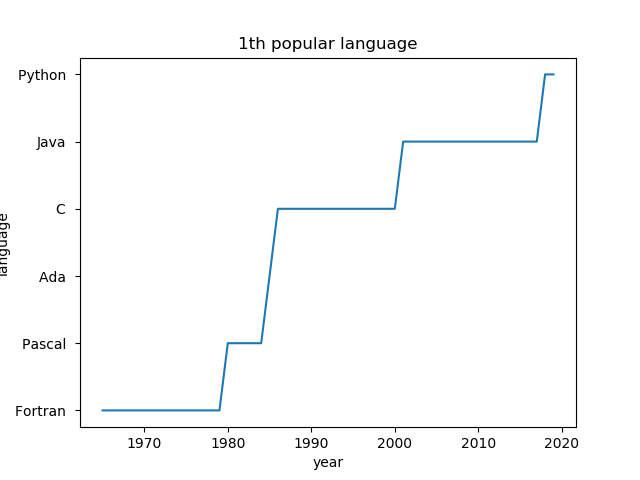
**运行结果：**

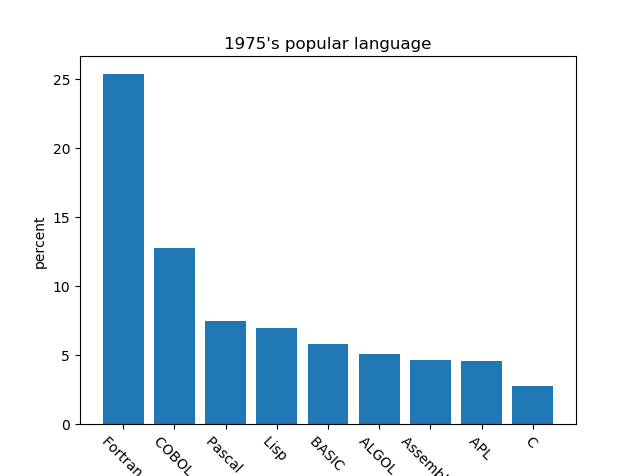
主机配置：LENOVO RESCUER R720-15IKBN, CPU:INTER I5-7300HQ, RAM:8GB

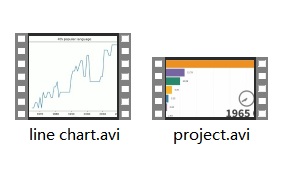
运行环境：WIN10 开发环境：Anaconda python 3.73

输入文件：testf.csv

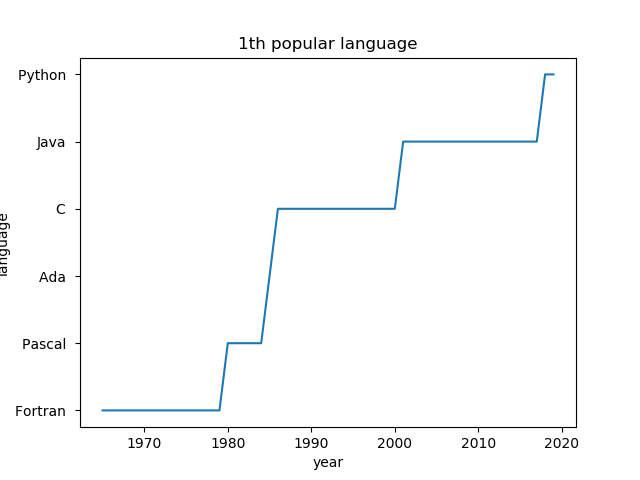
输出文件：line chart.avi(显示第k名流行的语言的变化情况), histogram.avi(反应每年的变化情况)







1. **对数据进行分析**



通过参考绘制出的每年最流行的语言变化图（如图），可以看出编程语言的变化趋势。

Fortran在上世纪70年代一直最流行。查看资料可知它是世界上最早出现的计算机高级程序设计语言，广泛应用于科学和工程计算领域。而结合当时计算机并不十分流行，并且主要用于这些领域，其流行度可想而知。

之后的80年代，Pascal取代了Fortran成为最流行的语言。Pascal的名称是为了纪念十七世纪法国著名哲学家和数学家Blaise Pascal而来的，它由[瑞士](https://baike.baidu.com/item/%E7%91%9E%E5%A3%AB" \t "https://baike.baidu.com/item/Pascal/_blank)[Niklaus Wirth](https://baike.baidu.com/item/Niklaus Wirth" \t "https://baike.baidu.com/item/Pascal/_blank)教授于六十年代末设计并创立的。[Pascal语言](https://baike.baidu.com/item/Pascal%E8%AF%AD%E8%A8%80" \t "https://baike.baidu.com/item/Pascal/_blank)语法严谨，层次分明，程序易写，可读性强，是第一个结构化编程语言。Pascal语言广泛用于各种软件，程序分为名称（program后自拟）、设置（var后规定）、开始（begin)、程序（正文）、读取（read/read ln）、结束（end），结构层次强，严谨而又紧密。由于其结构化的特点，被人们接受是理所当然的。

之后Ada短暂的登场。在20世纪70年代，[美国国防部](https://baike.baidu.com/item/%E7%BE%8E%E5%9B%BD%E5%9B%BD%E9%98%B2%E9%83%A8" \t "https://baike.baidu.com/item/Ada/_blank)(DoD)关切到其[嵌入式计算机系统](https://baike.baidu.com/item/%E5%B5%8C%E5%85%A5%E5%BC%8F%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%B3%BB%E7%BB%9F" \t "https://baike.baidu.com/item/Ada/_blank)项目中使用的众多不同编程语言，其中许多是过时或依赖于硬件，而其中更没有一个可以支援安全的模块化编程。于是在1975年成立高级语言工作组（HOLWG），意图创造适合于美英两国国防部需要的[编程语言](https://baike.baidu.com/item/%E7%BC%96%E7%A8%8B%E8%AF%AD%E8%A8%80" \t "https://baike.baidu.com/item/Ada/_blank)，以及精简现有编程语言数量。该项目最终成就了Ada语言。它被誉为第四代计算机语言的成功代表。与其他流行的程序设计语言不同，它不仅体现了许多现代软件的开发原理，而且将这些原理付诸实现。因此，Ada语言的使用可大大改善软件系统的 清晰性， 可靠性， 有效性， 可维护性。通过这些描述可见Ada比起之前的语言优势明显。

之后C语言取代Ada并占据霸主地位。C语言是一门面向过程的、抽象化的通用程序设计语言，广泛应用于底层开发。C语言能以简易的方式[编译](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91/1258343" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)、处理低级[存储器](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E5%99%A8/1583185" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)。C语言是仅产生少量的[机器语言](https://baike.baidu.com/item/%E6%9C%BA%E5%99%A8%E8%AF%AD%E8%A8%80/2019225" \t "https://baike.baidu.com/item/c%E8%AF%AD%E8%A8%80/_blank)以及不需要任何运行环境支持便能运行的高效率程序设计语言。由于其高效率与过程化特性，方便硬件编程（嵌入式）的特性，流行是必然的事情。

再之后是java统治的10年。Java是一门[面向对象](https://baike.baidu.com/item/%E9%9D%A2%E5%90%91%E5%AF%B9%E8%B1%A1" \t "https://baike.baidu.com/item/java/_blank)编程语言，不仅吸收了[C++](https://baike.baidu.com/item/C++" \t "https://baike.baidu.com/item/java/_blank)语言的各种优点，还摒弃了C++里难以理解的[多继承](https://baike.baidu.com/item/%E5%A4%9A%E7%BB%A7%E6%89%BF" \t "https://baike.baidu.com/item/java/_blank)、[指针](https://baike.baidu.com/item/%E6%8C%87%E9%92%88/2878304" \t "https://baike.baidu.com/item/java/_blank)等概念，因此Java语言具有功能强大和简单易用两个特征。Java语言作为静态面向对象编程语言的代表，极好地实现了面向对象理论，允许程序员以优雅的思维方式进行复杂的编程。Java具有简单性、面向对象、[分布式](https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F/19276232" \t "https://baike.baidu.com/item/java/_blank)、[健壮性](https://baike.baidu.com/item/%E5%81%A5%E5%A3%AE%E6%80%A7/4430133" \t "https://baike.baidu.com/item/java/_blank)、[安全性](https://baike.baidu.com/item/%E5%AE%89%E5%85%A8%E6%80%A7/7664678" \t "https://baike.baidu.com/item/java/_blank)、平台独立与可移植性、[多线程](https://baike.baidu.com/item/%E5%A4%9A%E7%BA%BF%E7%A8%8B/1190404" \t "https://baike.baidu.com/item/java/_blank)、动态性等特点 [2]  。Java可以编写[桌面应用程序](https://baike.baidu.com/item/%E6%A1%8C%E9%9D%A2%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F/2331979" \t "https://baike.baidu.com/item/java/_blank)、[Web应用程序](https://baike.baidu.com/item/Web%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F" \t "https://baike.baidu.com/item/java/_blank)、[分布式系统](https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F%E7%B3%BB%E7%BB%9F/4905336" \t "https://baike.baidu.com/item/java/_blank)和[嵌入式系统](https://baike.baidu.com/item/%E5%B5%8C%E5%85%A5%E5%BC%8F%E7%B3%BB%E7%BB%9F/186978" \t "https://baike.baidu.com/item/java/_blank)应用程序等。其面向对象与简单的特征，以及广泛的用途，使得许多企业使用其开发应用程序，因此流行了很长时间。

再之后python占据了流行的宝座。python相比别的高级语言集成度更高，拥有许多库可以实现所需功能（而其他语言常常需要自己编写）。同时其上手简单，适合新手学习。同时python在人工智能领域的应用广泛，而最近深度学习的 火热推动了它的发展。

总体看来，语言的流行程度主要取决于当时人们的需求，比如嵌入式火热时C流行，而深度学习火热时python流行度就很高。当然也与使用难度有关，比如结构化的Pascal登场，比起Fortran乃至汇编语言理所当然的更受到欢迎，python取代java也是如此。另外就是技术的发展决定了流行的语言，比如更优秀的C未登场前，人们只能选择当时已经有的比较难懂的语言。

最后分析一下2019年的情况。如图所示，python由于其应用的简单和广泛性使用者最多，而之后由于现在移动端和网页端程序的需求增长，JS和PHP等语言也十分靠前。而传统的面向对象语言仍然有很多应用，所以紧随其后。

