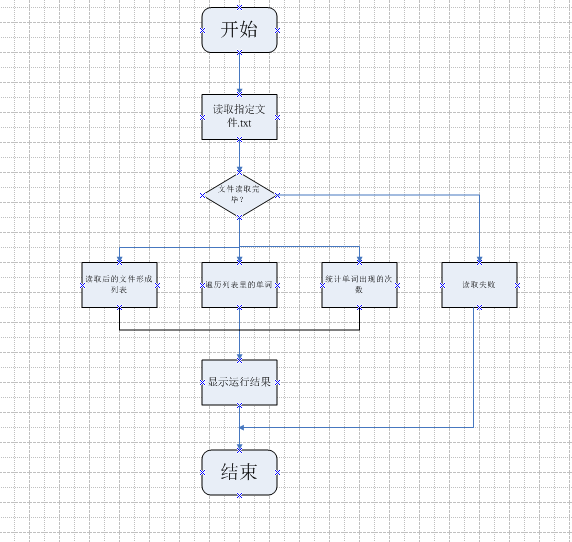
1. 面向对象的设计

通过对问题对象的分析，为了要实现词频统计的输出结果，要具体完成以下几方面的设计：

1. 总体设计——概要设计以及如何实现目标系统

总体设计流程图：



1. 方法设计：生成词频统计界面，判断是否统计成功



1. 详细设计——总体设计的具体化

4、具体代码实现

（1）人机界面的设计

对于程序读取文本并显示出结果之后，界面上所显示的内容，具体算法（代码）实现如下：



（2）总体程序代码

def openDir(self):

# 选择要打开的文件夹路径

self.dirPath.set(filedialog.askdirectory() + "/")

def doCalculate(self):

# 统计该文件下的词频

self.countDocWordsInDir(self.dirPath.get())

# 获得单个文档词频统计的计数器对象Counter

def getFileCounter(self, filepath):

# 拿到文本

file = open(filepath, "r", encoding="gbk")

text = file.read()

# 英文单词的正则表达式

pattern = "[A-Za-z]+"

# 从text中找出所有英文单词形成列表

wordlist = re.findall(pattern, text)

# wordlist的基础上创建Counter对象

counter = Counter(wordlist)

# 返回单个文件的计数器对象

return counter

# 统计一个文件夹下所有文档的一揽子词频

def countDocWordsInDir(self, dirpath):

# 罗列一个文件下的所有文件(含文件夹)

flist = os.listdir(dirpath)

# 创建空的计数器对象，统计对象通过counter对象的加法操作逐步扩大

mcounter = Counter()

# 遍历文件夹下的所有文件

for name in flist:

# 对文档文件进行统计操作

if name.endswith(".txt"):

docpath = dirpath + name # 获得文档文件的路径

# 获得文档的计数器对象

counter = self.getFileCounter(docpath)

# 计数器对象进行加法操作，扩大统计样本

mcounter += counter

# 得到结果

resList = mcounter.most\_common()

#print(resList)

# 显示结果

for item in resList:

self.retText.insert(END, str(item) + "\n")

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

DirectionDocWordCounter()

Pass

1. 测试结果如下：

