词频统计软件设计

一、词频统计简介

词频分析(Word Frequency Analysis)是对文献正文中重要词汇出现的次数进行统计与分析，是文本挖掘的重要手段。它是文献计量学中传统的和具有代表性的一种内容分析方法，基本原理是通过词出现频次多少的变化，来确定热点及其变化趋势。来确定文本文件中单词出现的频率。

二、设计思想

1、因为功能相对简单，采用python语言直接进行编写。

2、项目包含的统计功能利用定义的结构体分别对单词和次数进行统计。

3、以字符串的形式读取单词，并对其中的每个字符进行标点分析。

4、统计完成后采用降序的方式对次数进行排序。

5、将整个统计结果输出。三、数独软件生命周期设计

1. 问题定义：要解决什么问题

设计出词频统计软件。

1. 可行性研究：探索是否值得去解决

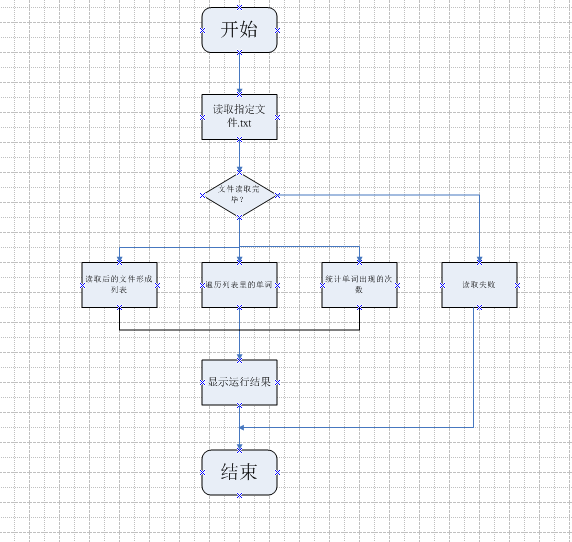
人物力可行，值得解决。

1. 需求分析：确定目标系统必须具备哪些功能

实现可视化、读取添加的文本文件、判断文本格式、在可视化界面输出结果

1. 总体设计：概要设计，如何实现目标系统

实现流程图：



1. 方法设计：生成词频统计界面、判断是否读取统计成功



1. 详细设计：把总体设计的抽象化，类似工程蓝图



1. 编码和单元测试：

import os # 操作路径

import re # 用于描述英文单词构成

from collections import Counter # 字符统计器

from tkinter import filedialog # 文件路径对话框

from tkinter import \* # GUI

class DirectionDocWordCounter:

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

# 定义排版边距

marginx = 5

marginy = 5

# 窗口和标题

window = Tk()

window.title("词频统计器")

# 使用StringVar对象动态保管打开的文件夹位置

self.dirPath = StringVar()

self.dirPath.set("选择要打开的文件夹")

# 第一行控件打包在一个面板中

frame1 = Frame()

frame1.pack(padx=marginx, pady=(marginy, 0))

# 打包地址输入框、地址选择按钮在面板1中

Entry(frame1, width=50, textvariable=self.dirPath).pack(side=LEFT)

Button(frame1, text="选择文件夹", command=self.openDir).pack(side=LEFT, padx=(marginx, 0))

# 词频统计按钮

Button(width=60, text="统计文件词频", command=self.doCalculate).pack(padx=marginx, pady=(10,0))

# 面板2用于盛放文本域和滚动条

frame2 = Frame()

frame2.pack(padx=marginx, pady=(marginy, marginy),expand=True,fill=Y)

# 打包文本域和滚动条在面板2中

self.retText = Text(frame2, width=58, height=20, bg="white")

scrollbar = Scrollbar(frame2,orient=VERTICAL,bg="black")

self.retText.pack(side=LEFT,expand=True,fill=Y)

scrollbar.pack(side=LEFT,expand=True,fill=Y)

# 双向关联文本域和滚动条

self.retText.config(yscrollcommand=scrollbar.set)

scrollbar.config(command=self.retText.yview)

# 消息循环

window.mainloop()

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")

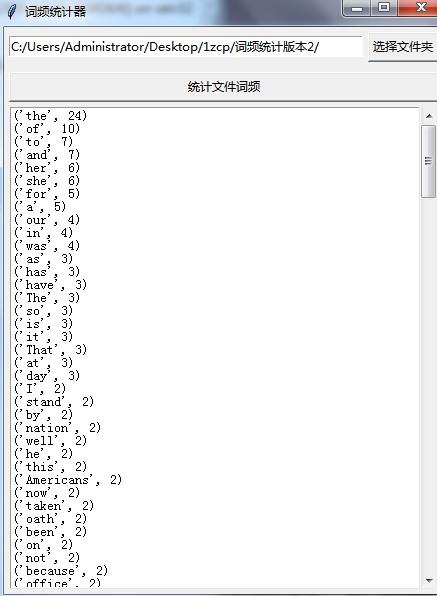
if \_\_name\_\_ == '\_\_main\_\_':

DirectionDocWordCounter()

Pass

（七）综合测试





1. 软件维护

可视化窗口不够美观，功能不够完全，需要的东西还欠缺很多代码结构需要优化等……