# 

# 软件工程报告

## 一、实验目的

1. 练习基本的编程能力；
2. 学习git分布式源代码管理工具的使用；

## 二、实验环境

硬件：个人笔记本，GPU：1660ti，CPU：Intel i7-9750H

软件：Jupyter Notebook(python3.7环境下)，anaconda，GitHub

## 三、实验要求

1. 注册GitHub账号，创建仓库；
2. 个人独立完成；
3. 提交GitHub仓库地址；

## 四、实验内容

### 用户需求：

英语的26 个字母的频率在一本小说中是如何分布的？某类型文章中常出现的单词是什么？某作家最常用的词汇是什么？《哈利波特》 中最常用的短语是什么，等等。 我们就写一些程序来解决这个问题，满足一下我们的好奇心。

### 第0步：

**需求**：输出某个英文文本文件中 26 字母出现的频率，由高到低排列，并显示字母出现的百分比，

命令行参数是：

main.exe -c <file name>

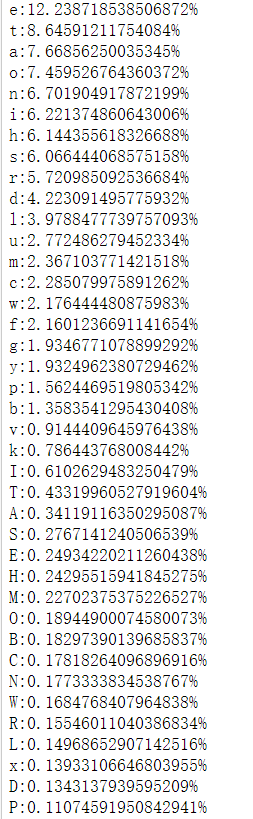
字母频率 = 这个字母出现的次数 / （所有A-Z，a-z字母出现的总数）

如果两个字母出现的频率一样，那么就按照字典序排列。 如果 S 和 T 出现频率都是 10.21%， 那么， S 要排在T 的前面。

**功能完成情况**：完成

**测试数据集**：完整测试集

**测试结果（部分）**：



**运行时间**：24.9898955s

### 第一步：

输出单个文件中的前 N 个最常出现的英语单词。

作用：一个用于统计文本文件中的英语单词出现频率的控制台程序  
单词：以英文字母开头，由英文字母和字母数字符号组成的字符串视为一个单词。单词以分隔符分割且不区分大小写。在输出时，所有单词都用小写字符表示。

英文字母：A-Z，a-z  
字母数字符号：A-Z，a-z，0-9  
分割符：空格,非字母数字符号 例：good123是一个单词，123good不是一个单词。good，Good和GOOD是同一个单词

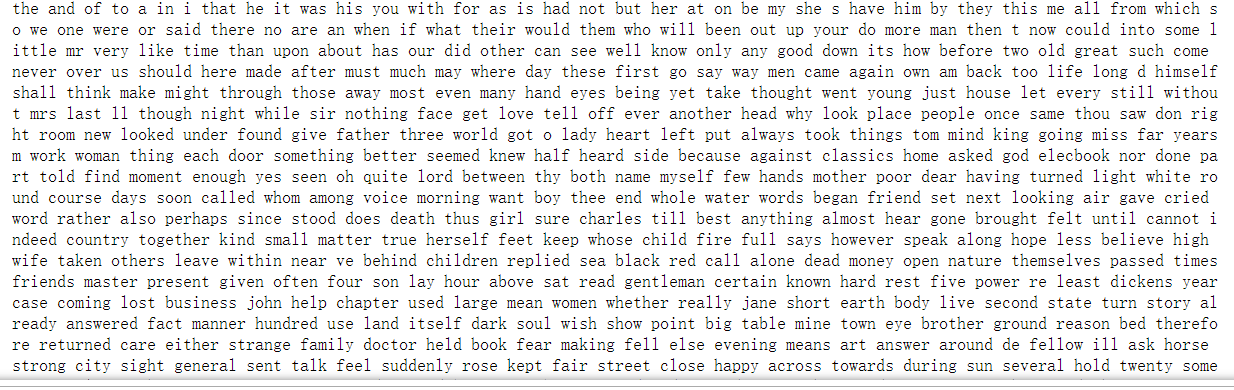
#### 功能1：main.exe -f <file name>

输出文件中所有不重复的单词，按照出现次数由多到少排列，出现次数同样多的，以字典序排列。

**功能完成情况**：完成

**测试数据集**：完整测试集

**测试结果（部分）**：



**运行时间**：37.317671700000005s

#### 功能2：

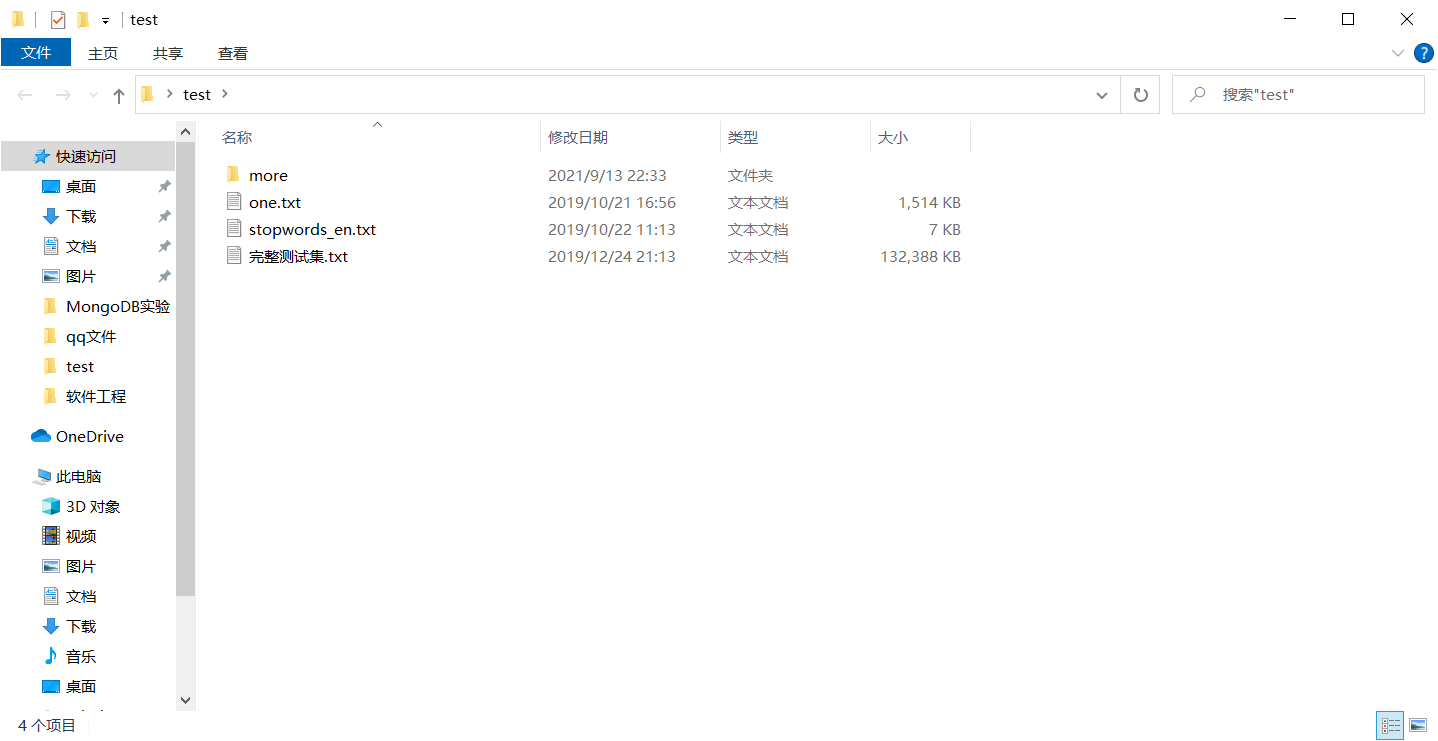
##### main.exe -d <directory name>

指定文件目录，对目录下每一个文件执行 main.exe -f <file name> 的操作。

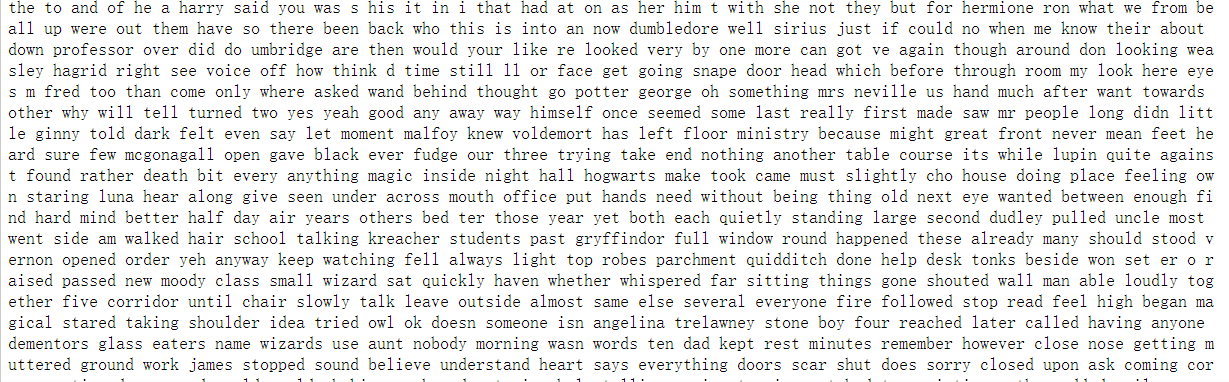
**功能完成情况**：完成

**测试数据集**：

一个文件夹，里面包含内容如下（more文件夹下放入one.txt与stopwords\_en.txt文件）：



**测试结果（部分）**：



**运行时间**：40.469365999999994s

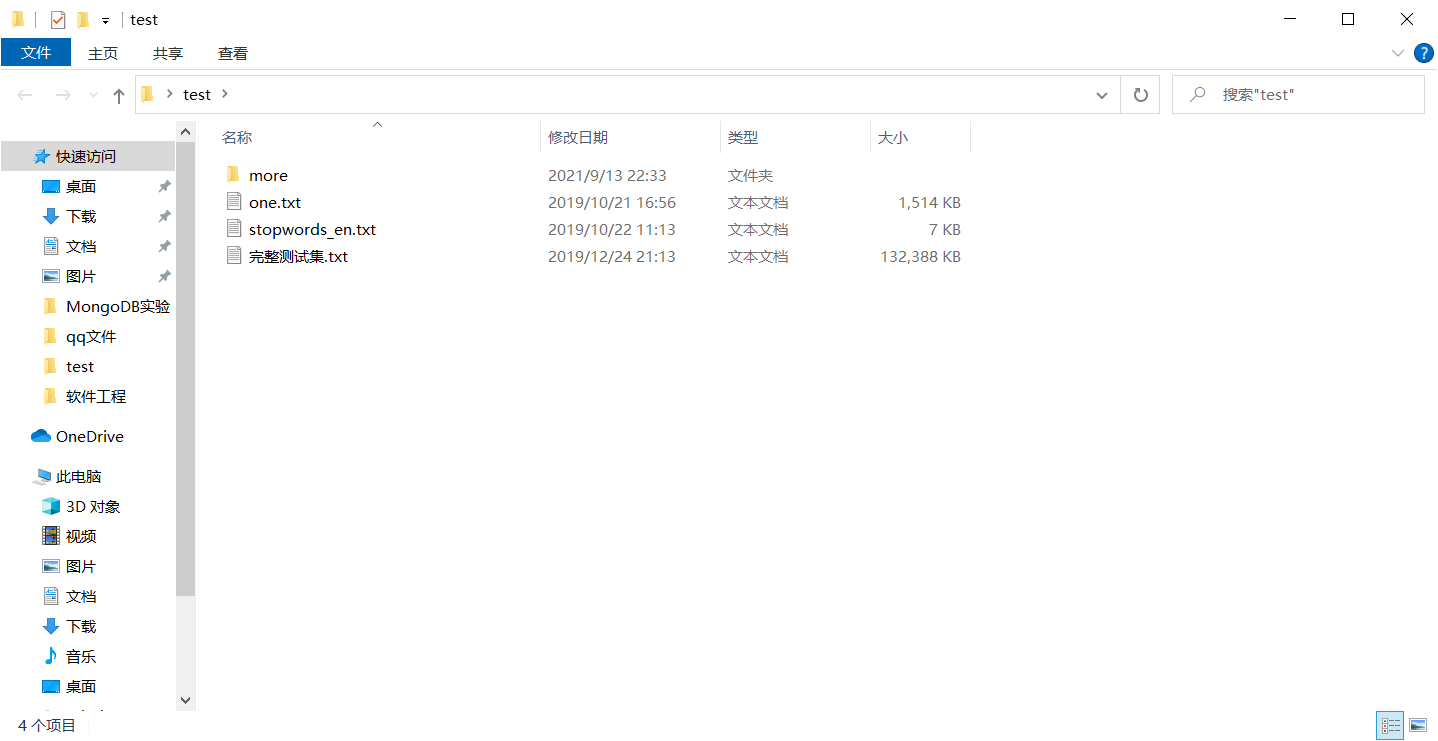
##### main.exe -d -s <directory name>

同上， 但是会递归遍历目录下的所有子目录。

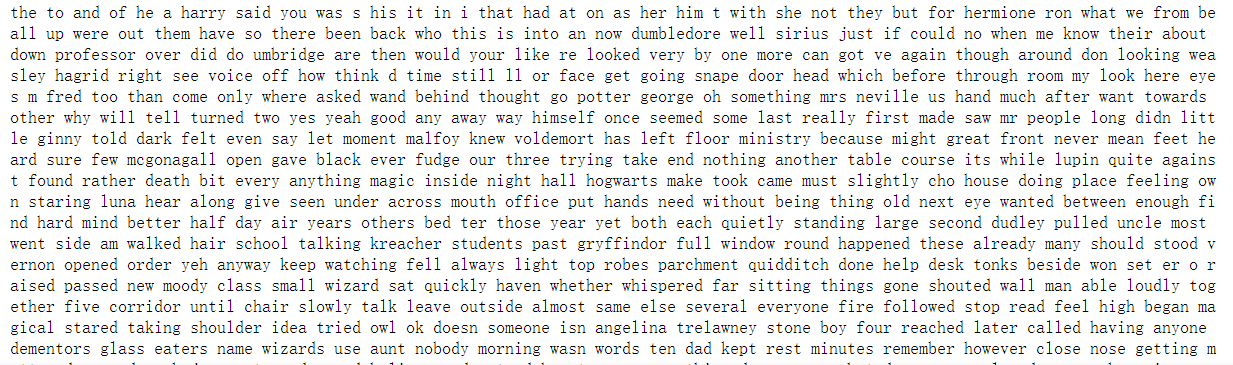
**功能完成情况**：完成

**测试数据集**：

一个文件夹，里面包含内容如下（more文件夹下放入one.txt与stopwords\_en.txt文件）：



**测试结果（部分）**：



**运行时间**：41.96293920000001s

#### 功能3：main.exe -n <file name> number

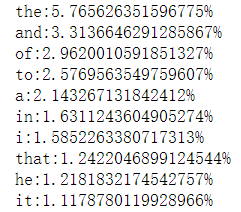
支持 -n 参数，输出出现次数最多的前 n 个单词， 例如， -n 10 就是输出最常出现单词的前 10 名。 当没有指明数量的时候，我们默认列出所有单词的频率。

##### 指定number时：

**功能完成情况**：完成

**测试数据集**：完整测试集，number=10

**测试结果（完整）**：



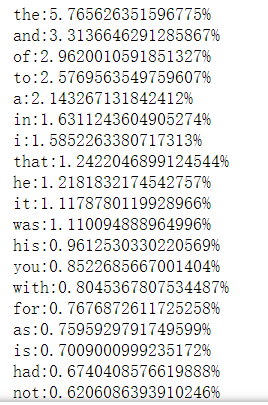
**运行时间**：20.628015900000037s

##### 未指定number时：

**功能完成情况**：完成

**测试数据集**：完整测试集

**测试结果（部分）**：



**运行时间**：38.065249400000084s

### 第二步

支持 stop words

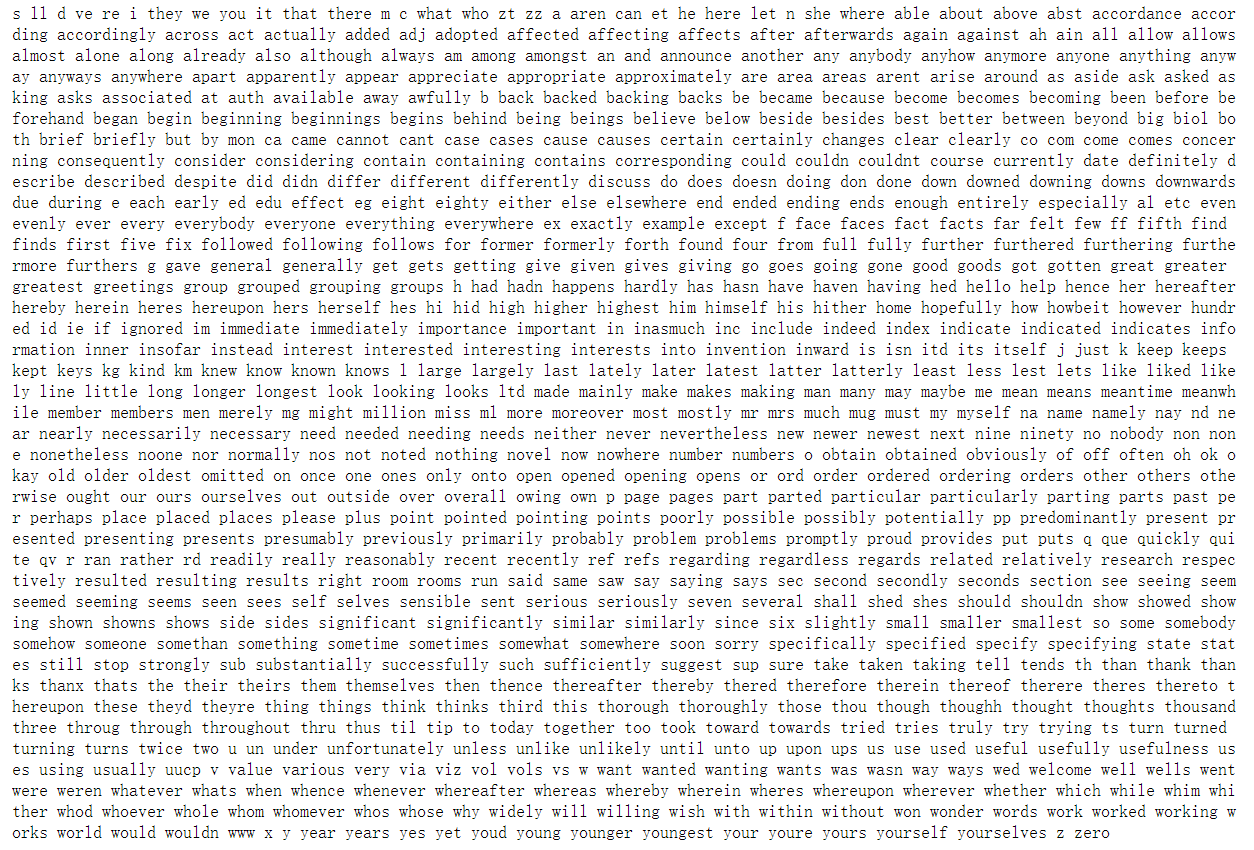
我们从第一步的结果看出，在一本小说里， 频率出现最高的单词一般都是 "a", "it", "the", "and", "this", 这些词， 我们并不感兴趣. 我们可以做一个 stop word 文件 （停词表）， 在统计词汇的时候，跳过这些词。 我们把这个文件叫 "stopwords.txt" file.

#### 功能 4：main.exe -x <stopwordfile name> -f <file name>

**功能完成情况**：完成

**测试数据集**：完整测试集

**测试结果（完整）**：



**运行时间**：19.63262900000001s

### 第三步:

我们想看看常用的短语是什么， 怎么办呢？

先定义短语："两个或多个英语单词， 它们之间只有空格分隔". 请看下面的例子：

hello world //这是一个短语

hello, world //这不是一个短语

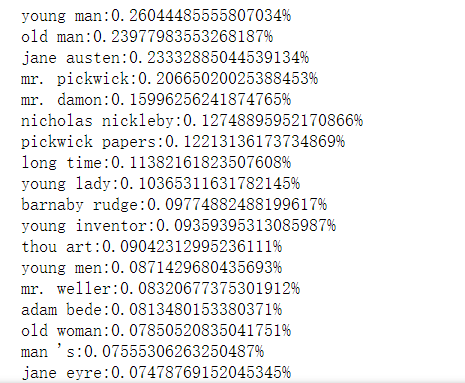
#### 功能 5：main.exe -p <file name> number

参数 number说明要输出多少个词的短语，并按照出现频率排列。同一频率的词组， 按照字典序来排列。

**功能完成情况**：完成，基于nlp实现，实现了对语义的分割

**测试数据集**：完整测试集，number=2

**测试结果（部分）**：



**运行时间**：609.0092457000001s

### 第四步:

把动词形态都统一之后再计数。

我们想找到常用的单词和短语，但是发现英语动词经常有时态和语态的变化，导致同一个词，同一个短语却被认为是不同的。 怎么解决这个问题呢？

假设我们有这样一个文本文件，这个文件的每一行都是这样构成：

动词原型 动词变形1 动词变形2...

词之间用空格分开。

e.g. 动词 TAKE 有下面的各种变形

take takes took taken taking

我们希望在实现上面的各种功能的时候，有一个选项， 就是把动词的各种变形都归为它的原型来统计。

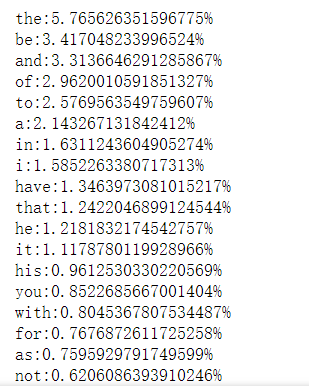
#### 功能 6：wf.exe -v <file name>

把动词的各种变形都归为它的原型来统计。

**功能完成情况**：完成，基于nlp实现，实现了英语单词的还原

**测试数据集**：完整测试集

**测试结果（部分）**：



**运行时间**：38.91084660000001s

## 五、实验结论

通过这次实验，我学会了git工具的使用；

锻炼了个人编程能力。

## 六、仓库地址

<https://github.com/Osilly/SE-Personal-programming.git>