链滴

单词出现频率分析小程序

作者：kuuyee
原文链接：https://ld246.com/article/1446159026649
来源网站：链滴

需求

前两天在一个孩子英语学习的家长QQ群里，有家长问道《夏洛的网》的单词量是多少，有的家长用Excel分析出了结果，但又有人问如何按各种要求排序，这个群里没有Excel高手，所以我写了这个单词分的小程序，主要是为golang学习练手。下面是程序的功能需求：

- 读取给定文本文件中的单词
- 分析文本文件中单词的出现频率
- 按照出现频率从高到低排序
- 出现频率相同的单词又按照首字母排序

实现

实现过程中遇到很多排序的算法问题，本人写程序经验欠缺，想到的实现方式可能**很不优雅**，也希望**神**们给予**指正**。

源码

```
package main

import (
    "bufio"
    "fmt"
    "io"
    "os"
    "regexp"
    "sort"
    "strconv"
    "strings"
)

func main() {
    file, err := os.Open("CharlotteWeb.txt")
    if err != nil {
        fmt.Fprintf(os.Stderr, "打开文件报错：%s\n", err)
    }

    charlotteWeb := make([]string, 0, 40000)

    // 匹配英文单词的正则
    r, _ := regexp.Compile("([\\w]+(\\'|\\-)*[\\w]+)|([\\w]+)")

    bufferReader := bufio.NewReader(file)

    for {
        // 以换行符位单位读取文件内容
        next, err := bufferReader.ReadString('\n')
        if err != nil {
            // 判断是否为文件尾部
```

```go
            if err != io.EOF {
                fmt.Fprintf(os.Stderr, "读取报错：%s\n", err)
            }
            break
        }
        // 匹配英文单词
        line := r.FindAllString(next, -1)

        if len(line) != 0 {
            // 全部转换为小写
            for i, v := range line {
                line[i] = strings.ToLower(v)
            }
            charlotteWeb = append(charlotteWeb, line...)
        }
    }

    fmt.Println("统计结果：")
    fmt.Printf("\n")
    fmt.Println("夏洛的网单词总数：", len(charlotteWeb))

    // 按照首写字母排序，位后面去重做准备
    sort.Strings(charlotteWeb)

    fmt.Printf("\n")

    // 去重
    ret := removeDuplicatesAndEmpty(charlotteWeb)

    fmt.Println("去重后单词总数：", len(ret))

    // 词频统计，并写入文件
    frequencyStatistics(charlotteWeb, ret)

}

// 利用对比相邻单词是否一样的原理来去重
func removeDuplicatesAndEmpty(a []string) (ret []string) {
    a_len := len(a)
    for i := 0; i < a_len; i++ {
        if (i > 1 && a[i-1] == a[i]) || len(a[i]) == 0 {
            continue
        }
        ret = append(ret, a[i])
    }
    return
}

// 词频统计函数，并把结果写入文件
// charlotteWeb: 夏洛的网所有单词切片
// ret 夏洛的网单词去重后的切片
func frequencyStatistics(charlotteWeb, ret []string) {

    // 新建词频统计写入文件
```

```
    wsFile, err := os.OpenFile("word_statistics.txt", os.O_CREATE|os.O_APPEND|os.O_RDWR, 0
60)
    if err != nil {
        fmt.Fprintf(os.Stderr, "统计文件打开错误：", err)
    }
    defer wsFile.Close()

    // 存放去重后每个单词的出现频率
    sortString := make([]int, 0, 4000)
    // 存放 map[单词]出现频率
    mapF := make(map[string]int, 4000)

    // 对于每个单词计算出现频率，然后存入map
    for _, rv := range ret {
        k := 0
        for _, cv := range charlotteWeb {
            if rv == cv {
                k++
            }
        }
        sortString = append(sortString, k)
        mapF[rv] = k
        //bufW.WriteString(rv + "    = " + strconv.Itoa(k) + "\r\n")
    }
    //fmt.Println(len(mapF))

    //bufW.Flush()

    // 对于保存词频的切片去重
    retInt := make([]int, 0, 1000)
    for i := len(charlotteWeb) - 1; i >= 0; i-- {
        for _, v := range sortString {
            if i == v {
                retInt = append(retInt, i)
                break
            }

        }
    }

    // 排序去重频率值
    sort.Ints(retInt)
    fmt.Printf("\n")
    fmt.Printf("全部出现的频率：%v\n", retInt)

    // 使用带缓冲的写入
    bufW := bufio.NewWriter(wsFile)

    // 判断map的值是否为频率值(频率值以排序，所以按排序写入)，然后写入文件，
    for i := len(retInt) - 1; i >= 0; i-- {
        // 用于临时保存相同频率的单词
        tempSort := make([]string, 0, 2000)
        for k, v := range mapF {
            if retInt[i] == v {
```

```
            tempSort = append(tempSort, k)
        }
    }

    // 对相同频率的单词以首写字母排序
    sort.Strings(tempSort)

    // 最后写入文件
    for _, v := range tempSort {
        bufW.WriteString(v + "    = " + strconv.Itoa(retInt[i]) + "\r\n")
    }
  }
  bufW.Flush()
}
```

输入结果

```
the    = 1941
and    = 1173
to    = 804
a    = 784
he    = 559
of    = 557
in    = 457
i    = 417

...

has    = 48
how    = 47
some    = 47
them    = 47
could    = 46
heard    = 46
fair    = 45
egg    = 43
never    = 43
any    = 42
come    = 42
we    = 42
who    = 41
again    = 40
```