lttzz / **Course-Design**

Branch: **master** **Course-Design** / T6 / **6.cpp** Find file Copy path

lttzz 词频统计(bug再次修复) 7e903d4 6 hours ago

1 contributor

381 lines (355 sloc) 10 KB

```
/*************************************************************************
    > File Name:        6.cpp
    > File Category: Course Design
    > Author:       lttzz
    > Mail:             3344517687@qq.com
 ************************************************************************/

//缺陷: 1)不规则动词复数,动词五态,s结尾的单词等可能会出现问题
//      2)C++等末尾为两个符号的会被误删符号

/*思路:
 *从文件中读取字符串,将其统一成小写字母后送到divideString()函数里,
 *字符串末尾的标点符号删除(如果有的话),处理完的结果以文件流的形式保存到本地备用
 *接着读取上一步处理后所得结果,进行二次处理.二次处理包括: 1)含数字单词变字母操作 2)复数变单数操作
 *上一步结果以键值对的形式存入到map中, 由于map内部本身就是按序存储的(key),故直接输出map即可实现按字典序输出每个单词及其出现次数
 *然后将键值对从map中取出来, 插入到vector中, 用sort()按词频降序排列后保存至本地并输出
*/
#include <iostream>
#include <fstream>
#include <map>
#include <vector>
#include <algorithm>
#include <iomanip>
#include <iterator>
using namespace std;

ifstream pin("./bbe.txt");
ofstream pout("./tmp.txt");
ifstream fin("./tmp.txt");
ofstream fout("./out.txt");

map<string, int> wordlist;
vector<pair<string,int> > Sort;
string temp_input, second_process;

void init();
void sortWordTime(void);
void selectMode(int op);
void divideString(string initial);
string processString(string simple);
bool checkAbnormal(string str);
void inputString(string after_process);
int cmp(const pair<string, int>& x, const pair<string, int>& y);


int main(void)
{
    init();
    int op;
    cin >> op;
    selectMode(op);
    return 0;
}


/**
 * 若文本出现单符号或者单数字, 返回false, 直接跳过此次读入的内容
 */
bool checkAbnormal(string str)
{
    if (str.length() == 1)
    {
```

```
        if (!(str[0] >= 'a' && str[0] <= 'z') || !(str[0] >= 'A' && str[0] <= 'Z'))
        {
            return false;
        }
    }
    return true;
}

void init()
{
    if(!fin || !fout || !pin || !pout)
    {
        cout << "Error: can't input or output file" << endl;
    }
    while(pin >> temp_input)
    {
        transform(temp_input.begin(), temp_input.end(), temp_input.begin(), ::tolower);     //转化为小写字母
        if (checkAbnormal(temp_input))
        {
            divideString(temp_input);
        }
    }

    pin.close();
    pout.close();

    while(fin >> second_process)
    {
        string last_word;
        last_word = processString(second_process);
        inputString(last_word);                                         //存入键值表
    }
    sortWordTime();
};

/**
 * 首先删去标点符号, 增加空格, 输出到pout文件中
 * 其次, 将一个字符串拆分成两个字符串
 * @param initial 待处理的原始字符串
 */
void divideString(string initial)
{
    if (initial[0] == '\'')     //删除字符串开头的引号(单引号和双引号)
    {
        initial.erase(0, initial.find_first_not_of('\''));
    }
    if (initial[0] == '\"')
    {
        initial.erase(0, initial.find_first_not_of('\"'));
    }

    //删除末尾符号,显然当末尾为符号时,若末尾前一位仍为符号,则末尾必为引号,都删除之
    int len = (int)initial.length();
    if (initial[len - 1] > 'z' || initial[len - 1] < 'a') {      //删除字符串后面的标点符号(如果有的话)
        initial.erase(len-1, 1);
        len--;
    }
    if (initial[len - 1] > 'z' || initial[len - 1] < 'a') {      //删除字符串后面的标点符号(如果有的话)
        initial.erase(len-1, 1);
        len--;
    }

    if(initial == "i'm")     //处理 ' 号分词问题
    {
        pout << "i am ";
    }
    else if(initial == "i'd")
    {
        pout << "i would ";
    }
    else if(initial == "i've")
    {
        pout << "i have ";
    }
```

```
    else if(initial == "wasn't")
    {
        pout << "was not ";
    }
    else if(initial == "isn't")
    {
        pout << "is not ";
    }
    else if(initial == "don't")
    {
        pout << "do not ";
    }
    else if(initial == "he's")
    {
        pout << "he is ";
    }
    else if(initial == "she's")
    {
        pout << "she is ";
    }
    else if(initial == "can't")
    {
        pout << "can not ";
    }
    else if(initial == "wouldn't")
    {
        pout << "would not ";
    }
    else if(initial == "you're")
    {
        pout << "you are";
    }
    else if(initial == "it's")
    {
        pout << "it is";
    }
    else if(initial == "i'll")
    {
        pout << "i will ";
    }
    else if(initial == "you'll")
    {
        pout << "you will ";
    }
    else if(initial == "he'll")
    {
        pout << "he will ";
    }
    else if(initial == "he'd")
    {
        pout << "he would ";
    }
    else if(initial == "she'd")
    {
        pout << "she would ";
    }
    else if(initial == "--")        //过滤破折号
    {
        return;
    }
    else if((initial[0] > 'z' || initial[0] < 'a') && (len == 1))      //过滤单符号
    {
        return;
    }
    else                                    //原样输出
    {
        pout << initial << ' ';
    }
}

/**
 * 含数字单词变字母操作,复数变单数操作(以's结尾的单词删除's)
 * ss结尾了解一下, was,is等等了解一下,2333
 */
```

```
211 string processString(string simple)
212 {
213     string ss[] = {"s", "dos", "is", "was", "as", "bus", "gas", "has", "his", "news", "odds", "plus", "status", "thank
214 
215     string last;
216     int simple_len = (int)simple.length();
217     if(simple == "1st")
218     {
219         last = "first";
220     }
221     else if(simple == "2nd")
222     {
223         last = "second";
224     }
225     else if(simple == "3rd")
226     {
227         last = "third";
228     }
229     else if(simple == "4th")
230     {
231         last = "forth";
232     }
233     else if(simple[simple_len-1] == 's')
234     {
235         if (simple[simple_len-2] == 's')        //双s结尾单词不是复数,故原样输出
236         {
237             return simple;
238         }
239 
240         for (it : ss)               //没辙了,手动输入白名单遍历特判吧
241         {
242             if (it == simple)
243             {
244                 return simple;
245             }
246         }
247 
248         if (simple[simple_len-2] == '\'')
249         {
250             simple.erase(simple_len-2, 1);
251             simple_len--;
252         }
253 
254         simple.erase(simple_len-1, 1);
255         simple_len--;
256         if(simple[simple_len-1] == 'e')
257         {
258             simple.erase(simple_len-1, 1);
259             simple_len--;
260             if(simple[simple_len-1] == 'i')
261             {
262                 simple.erase(simple_len-1, 1);
263                 simple += 'y';
264             }
265         }
266         last = simple;
267     }
268     else
269     {
270         last = simple;
271     }
272     return last;
273 }
274 
275 /**
276  * string->map
277  * @param after_process 第一次处理后的字串
278  */
279 void inputString(string after_process) {
280     auto it = wordlist.find(after_process);
281     if(it == wordlist.end())
282     {
283         wordlist.insert(pair<string, int>(after_process, 1));       //如果该单词没有出现过, 打包插入map, 并将值设为1
284     }
```

```
    else
    {
         wordlist[after_process]++;              //统计单词出现次数
    }
}

/**
 * sort()函数使用的cmp函数 使pair按单词出现次数多少排序
 */
int cmp(const pair<string, int>& x, const pair<string, int>& y) {
    return x.second > y.second;
}

/**
 * map->vector 并调用sort()排序
 */
void sortWordTime(void)
{
    for (auto it = wordlist.begin(); it != wordlist.end(); it++)
    {
        if (it->first == " ")
        {
            continue;
        }
        Sort.push_back(make_pair(it->first, it->second));
    }
    sort(Sort.begin(), Sort.end(), cmp);
}

/**
 * 选择模式 0--仅做处理后的文章,1-3对应于题目所做要求1-3
 * @param op [description]
 */
void selectMode(int op)
{
    if (0 == op)
    {

    }
    else if (1 == op)
    {
        cout << left << setw(20) << "Wordlist" << setw(20) << "Times" << "\n\n";
        fout << left << setw(20) << "Wordlist" << setw(20) << "Times" << "\n\n";
        for (auto it = wordlist.begin(); it != wordlist.end(); it++)
        {
            if(it->first[0] >= 'a' && it->first[0] <= 'z')        //如果键是单词 同时标准和文件输出键和值
            {
                cout << left << setw(20) << it->first << setw(20) << it->second << '\n';
                fout << left << setw(20) << it->first << setw(20) << it->second << '\n';
            }
        }
    }
    else if (2 == op)
    {
        int MAXN;
        cin >> MAXN;
        map<string, int> wd;
        cout << left << setw(20) << "Wordlist" << setw(20) << "Times" << "\n\n";
        fout << left << setw(20) << "Wordlist" << setw(20) << "Times" << "\n\n";
        for (vector<pair<string,int> >::iterator it = Sort.begin(); it != Sort.end(); it++)
        {
            if (MAXN == 0)
            {
                break;
            }
            if(it->first[0] >= 'a' && it->first[0] <= 'z')
            {
                wd.insert(pair<string, int>((*it).first, (*it).second));
                MAXN--;
            }
        }
        for (auto it = wd.begin(); it != wd.end(); it++)
        {
            if(it->first[0] >= 'a' && it->first[0] <= 'z')
```

```
359             {
360                 cout << left << setw(20) << it->first << setw(20) << it->second << '\n';
361                 fout << left << setw(20) << it->first << setw(20) << it->second << '\n';
362             }
363         }
364     }
365     else if (3 == op)
366     {
367         cout << left << setw(20) << "Wordlist" << setw(20) << "Times" << "\n\n";
368         fout << left << setw(20) << "Wordlist" << setw(20) << "Times" << "\n\n";
369         for (vector<pair<string,int> >::iterator it = Sort.begin(); it != Sort.end(); it++)
370         {
371             if(it->first[0] >= 'a' && it->first[0] <= 'z')
372             {
373                 cout << left << setw(20) << it->first << setw(20) << it->second << '\n';
374                 fout << left << setw(20) << it->first << setw(20) << it->second << '\n';
375             }
376         }
377     }
378
379     fin.close();
380     fout.close();
381 }
```