基于文本内容的新闻推荐与检索（1）实验报告

牛浩宇 软件 72 2017010729

1 实验目标与环境

本实验的目标是对新闻网页进行信息提取和分词，并为将来构建倒排文档、实现新闻查询及简单推荐打下基础。我在开发时使用的操作系统是 macOS Mojave 10.14.1，使用的 IDE 是 Clion，编程语言使用 C++11。经更换环境验证，程序在 Windows 10 + Visual Studio 环境下也可以正常运行。

2 抽象数据结构说明

本实验中，我主要实现了 CharString, CharStringLink, Stack, HashTable 几种数据结构。

2.1 CharString 类

一个本类对象包含一个 char* content 用于存储数据，一个 size_t contentLength 用于记录自定义字符串长度。

本类主要包含的函数有（拷贝、移动）构造函数，析构函数。对于 indexOf，要求传入子串和开始匹配的位置（默认从头开始），用 KMP 算法返回子串第一次出现的位置。对于 substring，要求传入子串的长度和截取开始的位置，返回 CharString*类型的子串。对于 concat，要求按顺序传入左子串和右子串，返回 CharString*为连接后的新字符串。此外，我还重载了赋值运算符 operator=，支持从 CharString 或 std::string 完成对内容的赋值。

2.2 CharStringLink 类

一个本类对象包含两个 CharStringNode* front/rear 用于存储链表头尾，一个 size_t _length 用于存储链表长度。对于每个 CharStringNode，都有*prior 和*next 两个指针，从而构成一个双向链表。

对于 add，要求传入添加节点的内容和节点位置；对于 remove，要求传入删除节点的位置；对于 search，要求传入元素内容。

2.3 Stack 类

本类使用了模版，其对象组成类似于 CharStringLink，即亦为双向链表式结构（便于操作）。对于 push_back()，要求传入节点内容；对于 pop/top/empty 则无要求。

2.4 HashTable 类

尽管课程内容尚未学到，但我利用自学知识成功完成了 hashTable 的搭建。注意哈希函数针对的是 GB2312 编码的汉字，因此输入文件需要符合该编码，否则将导致哈希表存

储效率降低。

哈希表采用开链法（separate chaining）搭建，其成员主要是__hashTable_node[hashSize]，其中__hashTable_node 表示哈希表的一个节点，hashSize 表示预先给定的哈希表大小。我目前实现了 insert, contained 和 size 三个函数，其中前两个要求输入 string 形式的中文，最好以 GB2312（或更近的国标如 GB18030）编码。

3 简要算法说明

本实验代码中主要用到的算法包括 KMP 算法，基于栈的标签匹配算法，以及正向最大匹配分词算法。

KMP 算法的思想在于通过预载时搭建的 next 表降低运行时开销，对此书上已有具体实现，不再赘述。标签匹配算法的流程类似附录中所给出的，不过针对复杂易错的现实情况（如多标签或少标签）进行了更多优化。正向最大匹配算法也是非常基础的算法，不过此处我利用了自己实现的哈希表，实现了更快的查找速度。

4 实验流程、操作说明与试验结果

由于我的开发环境是 macOS Mojave + Clion，因此把代码迁移到 Windows 10 + Visual Studio 的环境下仍然花了一点功夫。

我上交的是 Windows 10 迁移版。进行实验时，把需要解析的网页文件（统一以”.html”结尾）放在可执行程序下./input 路径里，把字典文件命名为 dictionary.dic，并放在可执行程序同路径下，即./dictionary.dic。然后，在可执行程序下新建一个./output 文件夹。注意，所有的文件建议为 GB2312 或 GB18030 编码，否则可能导致输出乱码。

然后，点击编译运行或直接打开 NewsFeedSystem.exe 可执行文件。待系统弹出的黑框自动关闭后，即可在./output 文件夹中找到按照实验要求输出的结果文档。试验结果已经在上传的范例中给出，不再赘述。

5 功能亮点

首先，我在自学的基础上成功实现了基于开链法的哈希表，相较传统的线性探测法有更高的效率。由于哈希表固有的特性，这使得我在查找字典时有了比遍历全链表更高的效率。

其次，我贯彻了面向对象的思想，较好地完成了接口的封装与底层代码复用。

最后，我优化后的标签匹配算法能有效过滤非文本及广告内容，从而可以提升匹配输出的效率。

6 实验体会

由于这是我第一次正式在类 Unix 系统下完成开发，因此中途遇到了很多意想不到的困难。最明显的比如 Unix 下文件默认即为 UTF-8 编码，而在 Windows 下文件为 Unicode 编码，因此我在完成代码的迁移时不得不把源代码编码全部进行转换。其次，Clion 和 Visual Studio 之间的差异让我明显地感觉到不同编译器的实现机制不同。对于同一段内存访问越界，Clion 可能自动帮你返回空值，而 Visual Studio 则会报错，让你自己进行修改。而且<Windows.h>和<io.h>这两个提取当前路径下所有文件所需的头文件，在 macOS 系统下并没有，故而需要复杂得多的办法来实现。

由于上述代码迁移过程中所遇到的未预料的困难，严重耽误了我完成大作业的进度，差点没能赶上 ddl。这也是我这次实验最重要的一点收获：不要总是卡 DDL，一定要给自己预留充足的时间。

最后，谢谢张力老师和助教的细心指导！

*按惯例：作业好简单啊，不够达到训练的目标，求 加量，求加难度