NLPIR 大数据搜索与挖掘说明文档

自然语言处理与信息检索共享平台

Natural Language Processing & Information Retrieval Sharing Platform

http://www.nlpir.org/

@ICTCLAS 张华平博士

2014-12

For the latest information about NLPIR, please visit Http://www.nlpir.org/

访问 http://www.nlpir.org/(自然语言处理与信息检索共享平台)，您可以获取 NLPIR 系统的最新版本，并欢迎您关注张华平博士的新浪微博 @ICTCLAS 张华平博士 交流。

下载地址：

http://yunpan.cn/cfSCSzNyTYQjW 提取码 4c49

目录

一、NLPIR 大数据搜索与挖掘开发平台简介

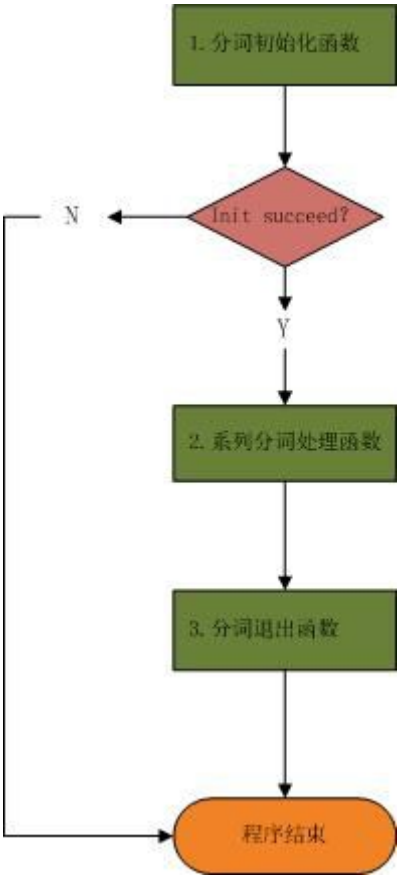
NLPIR 大数据搜索与挖掘开发平台指的是网络搜索、自然语言理解和文本挖掘的技术开发的基础工具集，开发平台由多个中间件组成，各个中间件 API 可以无缝地融合到客户的各类复杂应用系统之中，可兼容 Windows，Linux，FreeBSD 等不同操作系统，可以供 Java，C，C#等各类开发语言使用。具体中间件包括：分词组件(NLPIR)，关键词组件(KeyExtract)，聚类组件(Cluster) ，分类组件(Classifier) ，训练分类组件(DeepClassifier) ，文档抽取组件(DocExtractor)，摘要组件(Summary) ，去重组件(RedupRemover) ，情感分析组件(SentimentAnalysis) ，特定人物情感分析(Sentiment)，精准搜索组件(JZSearch)。

二、分词组件(NLPIR) 简要流程图及函数说明文档

汉语分词组件能对汉语语言进行拆分处理，是中文信息处理必备的核心部件。它综合了各家所长，采用条件随机场（Conditional Random Field,简称 CRF）模型，具备准确率高、速度快、可适应性强等优势。特色功能包括：切分粒度可调整，融合 20 余部行业专有词典，支持用户自定义词典等。

词性标注能对汉语语言进行词性的自动标注，它能够真正理解中文，自动根据语言环境将词语诸如“建设”标注为“名词”或“动词”。

2.1 分词组件函数流程图：



2.2 各个函数详细说明：

2.2.1 关键词初始化函数：

① Int NLPIR_Init(const char * sDataPath , int encode, const char*sLicenceCode);

功能：为分词系统初始化和准备环境；

备注：在进程中此函数必须在其他函数之前调用（仅需调用一次）

参数：sDataPath：Data 文件夹的路径，为空字符串时从项目根目录下开始寻找

encode：编码格式，具体的编码对照如下：

0：GBK

1：UTF8

2：BIG5

3：GBK，里面包含繁体字；

sLicenceCode：授权码，为空字符串就可以了

2.2.2 系列关键词处理函数：

① const char * NLPIR_ParagraphProcess(const char *sParagraph,int bPOStagged=1);

功能：对指定段落进行分词，并返回分词结果

备注：如果 Data 文件夹里面的 Configure.xml 中的标签 FinerSegCombined 为 true 那么是粗粒度分词+细粒度分词的混合模式，如果标签 FinerSegCombined 为 false，那么是粗粒度分词模式

参数：sParagraph：文本内容

bPOStagged：是否需要分词标记，0 表示不标记，1,表示需要标记，默认为 1；

② const char * NLPIR_GetLastErrorMsg();

功能：返回最近一次的错误信息

③ const result_t * NLPIR_ParagraphProcessA(const char * sParagraph,int *pResultCount,bool bUserDict=true);

功能：对指定段落进行分词，并返回分词结果

参数： sParagraph: The source paragraph

pResultCount: pointer to result vector size

④ int NLPIR_GetParagraphProcessAWordCount(const char *sParagraph);

功能：Get ProcessAWordCount, API for C#

Get word count and it helps us prepare the proper size buffer for result_t vector

参数：sParagraph: The source paragraph

⑤ void NLPIR_ParagraphProcessAW(int nCount,result_t * result);

功能：Process a paragraph, API for C#

参数：sParagraph: The source paragraph

result_t * result: pointer to result vector size, it is allocated by the invoker

⑥ double NLPIR_FileProcess(const char *sSourceFilename,const char * sResultFilename, int bPOStagged=1);

功能：Process a text file

参数：sSourceFilename: The source file name

sResultFilename: The result file name

bPOStagged:Judge whether need POS tagging, 0 for no tag;default:1

I.e.FileProcess("E:\\Sample\\Corpus_NewPOS\\199802_Org.txt","E:\\Sample\\Corpus_NewPOS\\199802_Org_cla.txt");

⑦ unsigned int NLPIR_ImportUserDict(const char *sFilename,bool bOverwrite=true);

功能：导入用户词典

备注：系统会自动处理重复词的问题

参数：sFilename - 用户词典的路径（请使用全路径）

bOverwrite - 是否删除原有的自定义用户词典，true：删除；false：不删除

⑧ unsigned int NLPIR_ImportKeyBlackList(const char *sFilename);

功能：导入黑名单，成功导入后，黑名单中出现的词，不会作为关键词出现。

参数：sFilename - 黑名单的路径（请使用权路径）

返回：成功提交的黑名单中单词的个数

⑨ int NLPIR_AddUserWord(const char *sWord);

功能：添加自定义单词

备注：系统会自动处理重复词的问题

参数：sWord--单词

⑩ int NLPIR_SaveTheUsrDic();

功能：Save dictionary to file

返回：1,true; 2,false

⑪ int NLPIR_DelUsrWord(const char *sWord);

功能：从用户词典中删除单词

返回：如果单词不存在就返回-1，否则返回单词在词典中的句柄

参数：sWord - 单词

⑫ double NLPIR_GetUniProb(const char *sWord);

功能： Get Unigram Probability

返回：success:

fail:

参数：sWord: input word

⑬ Int NLPIR_IsWord(const char *sWord);

功能：判断单词是否在核心词库中

返回：如果单词不存在就返回-1，否则返回单词在词典中的句柄

参数：sWord: 输入的单词

⑭ const char *NLPIR_GetKeyWords(const char *sLine,int nMaxKeyLimit,bool bWeightOut);

功能：获得关键词

返回：关键词

参数：sLine - 文本内容

nMaxKeyLimit - 关键词个数

bWeightOut - 是否显示权重

⑮ const char * NLPIR_GetFileKeyWords(const char *sFilename,int nMaxKeyLimit=50,bool bWeightOut=false);

功能：获得关键词

返回：关键词

参数：文本文件的路径（全路径）

nMaxKeyLimit - 关键词个数

bWeightOut - 是否显示权重

⑯ const char * NLPIR_GetNewWords(const char *sLine,int nMaxKeyLimit=50,bool bWeightOut=false);

功能：获得新词

返回：新词

参数：sLine--文本内容

nMaxKeyLimit--返回新词的个数

bWeightOut - 是否显示权重

⑰ Const char * NLPIR_GetFileNewWords(const char *sFilename,int nMaxKeyLimit=50,bool bWeightOut=false);

功能：获得新词

返回：新词

参数：sLine--文本文件的路径

nMaxKeyLimit--返回新词的个数

bWeightOut - 是否显示权重

⑱ unsigned long NLPIR_FingerPrint(const char *sLine);

功能：指纹提取函数

返回：文本的指纹码

参数：sLine - 文本内容

⑲ int NLPIR_SetPOSmap(int nPOSmap);

功能：select which pos map will use

返回值：0, failed; else, success

参数：nPOSmap--标注集，具体的标注集对应表如下：

ICT_POS_MAP_FIRST 1 //计算所一级标注集

ICT_POS_MAP_SECOND 0 //计算所二级标注集

PKU_POS_MAP_SECOND 2 //北大二级标注集

PKU_POS_MAP_FIRST 3 //北大一级标注集

⑳ CNLPIR* GetActiveInstance();

功能：获取可用的 CNLPIR 类，适用于多线程开发，先获取可用的 CNLP，再调用其中的功能

返回：可用的 CNLPIR 类

㉑ intNLPIR_NWI_Start();

功能：启动新词识别

返回：0：失败，其他：成功

㉒ int NLPIR_NWI_AddFile(const char *sFilename);

功能：往新词识别系统中添加待识别新词的文本文件，需要在运行 NLPIR_NWI_Start()之后，才有效

返回：0：失败，其他：成功

参数：*sFilename：文件名

㉓ int NLPIR_NWI_AddMem(const char *sText);

功能：往新词识别系统中添加一段待识别新词的内存，需要在运行 NLPIR_NWI_Start()之后，才有效

返回：0：失败，其他：成功

参数：sText -- 文本内容

㉔ int NLPIR_NWI_Complete();

功能：新词识别添加内容结束，需要在运行 NLPIR_NWI_Start()之后，才有效

返回：0：失败，其他：成功

㉕ const char * NLPIR_NWI_GetResult(bool bWeightOut=false);

功能：获取新词识别的结果，需要在运行 NLPIR_NWI_Complete()之后，才有效

返回：输出格式为【新词 1】【权重 1】【新词 2】【权重 2】 ...

参数：bWeightOut：是否需要输出每个新词的权重参数

㉖ unsigned int NLPIR_NWI_Result2UserDict();

功能：将新词识别结果导入到用户词典中，需要在运行 NLPIR_NWI_Complete()之后，才有效，如果需要将新词结果永久保存，建议在执行 NLPIR_SaveTheUsrDic

返回：新词结果数目

㉗ const char* NLPIR_FinerSegment(const char *sLine);

功能：当前的切分结果过大时，如“中华人民共和国”

需要执行该函数，将切分结果细分为“中华 人民 共和国”

细分粒度最大为三个汉字

返回：返回细粒度分词，如果不能细分，则返回为空字符串""

参数：sLine:输入的字符串

㉘ NLPIR_API const char * NLPIR_GetEngWordOrign(const char *sWord);

功能：获取各类英文单词的原型，考虑了过去分词、单复数等情况

返回：返回的词原型形式

driven->drive drives->drive drove-->drive

参数：sWord:输入的单词

㉙ NLPIR_API const char * NLPIR_WordFreqStat(const char *sText);

功能：获取输入文本的词，词性，频统计结果，按照词频大小排序

返回：返回的是词频统计结果形式如下：

张华平/nr/10#博士/n/9#分词/n/8

参数：sWord:输入的文本内容

㉚ NLPIR_API const char* NLPIR_FileWordFreqStat(const char *sFilename);

功能：获取输入文本的词，词性，频统计结果，按照词频大小排序

参数：sFilename 文本文件的全路径

返回： 返回的是词频统计结果形式如下：

张华平/nr/10#博士/n/9#分词/n/8

2.2.3 关键词退出函数：

① bool NLPIR_Exit();

功能：退出并释放系统资源；

2.3 分词组件 Java 调用示例

```
public class NlpirTest
{
Logger logger = Logger.getLogger(NlpirTest.class.getName());
static{
    boolean flag = CLibraryNlpir.Instance.NLPIR_Init("", 1, "0");
    if (flag) {
        System.out.println("nlpir 初始化成功");
    } else {
        System.out.println("nlpir 初始化失败：" +
CLibraryNlpir.Instance.NLPIR_GetLastErrorMsg());
    }
}
```

```
    @Test
    /**
     * 分词测试
     */
    public void testParticiple() {
        System.out.println(System.getProperty("java.library.path"));
            String sSrc = "李思雨进入了黑名单";
            String data = CLibraryNlpir.Instance.NLPIR_ParagraphProcess(sSrc, 1);
            System.out.println(data);
    }

    @Test
    /**
     * 细粒度分词测试
     */
    public void testFinerSegment(){
        String lenWords="阿拉德大陆";
        System.out.println("得到的返回值 2 为
-->"+CLibraryNlpir.Instance.NLPIR_FinerSegment(lenWords));

    }

    @Test
    /**
     * 关键词提取测试
     */
    public void testKeyWord() {
        String sSrc = "，全球旅行必备话中蒙友好合作，谈中国周边外交，论亚洲国家相处
之道，强调互尊互信、聚同化异、守望相助、合作共赢，共创中蒙关系发展新时代，共促
亚洲稳定繁荣";
        String data = CLibraryNlpir.Instance.NLPIR_GetKeyWords(sSrc, 10, false);
        logger.debug(data);
    }

    @Test
    /**
     * 新词提取测试
     */
    public void testNewWord() {
        String filePath = "test.txt";
        String sSrc = FileOperateUtils.getFileContent(filePath);
        logger.debug("文章内容为---->" + sSrc);
        String data = CLibraryNlpir.Instance.NLPIR_GetNewWords(sSrc, 1024, false);
        logger.debug(data);
```



```
    }
    @Test
    /**
     * 添加用户自定义词测试
     * 这个方法目前不能用
     */
    public void testAddUserWord() {
        int i = CLibraryNlpir.Instance.NLPIR_AddUserWord("阿拉德");
        if (1 == i) {
            logger.debug("用户词添加成功");
        } else {
            logger.debug("用户词添加失败");
        }
    }

    @Test
    /**
     * 添加用户词典测试
     */
    public void testImportUserDict() {
        int addDictNum = CLibraryNlpir.Instance.NLPIR_ImportUserDict("dict/dict.txt",
true);
        logger.debug("已经添加的用户自定义词个数（累加数）为：");
        System.out.println(addDictNum);
    }
    @Test
    /**
     * 添加用户黑名单词典测试
     */
    public void testImportKeyblacklist() {
        int addDictNum = CLibraryNlpir.Instance
                .NLPIR_ImportKeyBlackList("dict/keyblacklist.txt");
        logger.debug("已经添加的用户自定义词个数（累加数）为：");
        System.out.println(addDictNum);
    }
    @Test
    /**
     * 指纹提取测试
     */
    public void testFingerPrint(){
        String content="我爱北京天安门";
        long fingerPrint= CLibraryNlpir.Instance.NLPIR_FingerPrint(content);
        System.out.println(fingerPrint);
    }
```



```
    @Test
    /**
     * 是否在核心词典中测试
     */
    public void testIsWord(){
        String word="阿拉";
        System.out.println(CLibraryNlpir.Instance.NLPIR_IsWord(word));
    }

    @Test
    /**
     * 词性测试
     */
    public void testGetWordPOS(){
        String sWords="中华人民共和国";
        System.out.println("传入的词为-->"+sWords);
        String s=CLibraryNlpir.Instance.NLPIR_GetWordPOS(sWords);
        System.out.println("返回词性为-->"+s);
    }

    @Test
    public void testWordFreqStat() {
        String sSrc = "，全球旅行必备话中蒙友好合作，谈中国周边外交，论亚洲国家相处之道，强调互尊互信、聚同化异、守望相助、合作共赢，共创中蒙关系发展新时代，共促亚洲稳定繁荣";
        String result = CLibraryNlpir.Instance.NLPIR_WordFreqStat(sSrc);
        System.out.println(String.format("======词频结果如下：====\n%s", result));
    }

    @Test
    public void testFileWordFreqStat() {
        String filePath = "test.txt";
        String result = CLibraryNlpir.Instance.NLPIR_FileWordFreqStat(filePath);
        System.out.println(String.format("======词频结果如下：====\n%s", result));
    }

    @Test
    public void testNLPIR_GetEngWordOrign() {
        String word = "DRIVEN";
        String result = CLibraryNlpir.Instance.NLPIR_GetEngWordOrign(word);
        System.out.println(String.format("输入的单词：%s\n 单词的原型：%s", word, result));
    }
}
```


2.4 分词组件 C 调用示例

```
printf("按照单篇文章进行处理！\n");
        if(!NLPIR_Init("../../",GBK_CODE))//数据在当前路径下，默认为 GBK 编码的分词
        {
            printf("ICTCLAS INIT FAILED!\n");
            return ;
        }
        printf("Success Init!\n");
        char sInput[1000] = "先登机第一劝业银行【环球时报记者朱晓磊】据越南《青年
        日报》等媒体 2 月 18 日报道";
        clock_t lStart,lEnd;
        long lTime=0;
        int nLine=0;
        int nSize=strlen(sInput),nTotalSize=0;
        int i=1;
        double fTimeTotal=0.0,fTime=0.0;//Time cost
        double fSpeed,fSpeendMin=100000000.0;
        lStart=clock();//Record the time
        for (int i=0;i<10000;i++)
        {
            const char *pResult=NLPIR_ParagraphProcess(sInput);
        }
        lEnd=clock();//Record the time
        lTime=lEnd-lStart;
        fTime=(double)lTime/(double)CLOCKS_PER_SEC;//Time cost
        fSpeed=(double)nSize*10000/(double)fTime;
        printf("speed=%.2lfBytes/sec,length=%ld,time=%.6f\n",fSpeed,nSize*10000,fTime);
        NLPIR_Exit();
```


三、关键词组件(KeyExtract) 简要流程图及函数说明文档

关键词组件能够在全面把握文章的中心思想的基础上，提取出若干个代表文章语义内容的词汇或短语，相关结果可用于精化阅读、语义查询和快速匹配等。

采用基于语义的统计语言模型，所处理的文档不受行业领域限制，且能够识别出最新出现的新词语，所输出的词语可以配以权重。

文章关键词提取组件的主要特色在于：

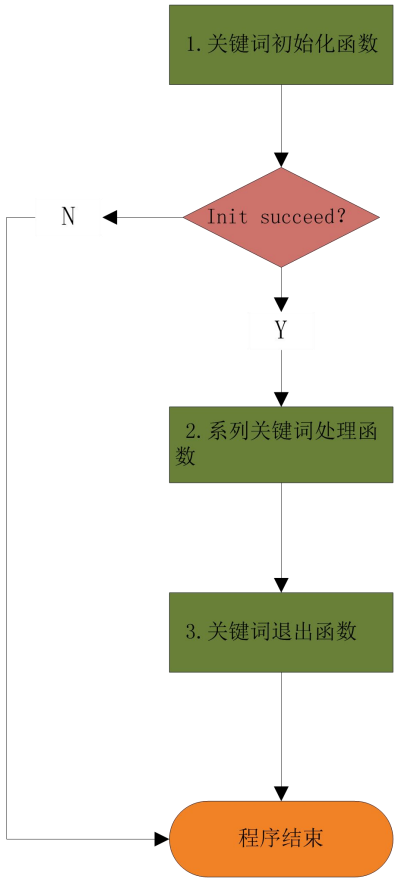
1、速度快：可以处理海量规模的网络文本数据，平均每小时处理至少 50 万篇文档；

2、处理精准：Top N 的分析结果往往能反映出该篇文章的主干特征；

3、精准排序：关键词按照影响权重排序，可以输出权重值；

4、开放式接口：文章关键词提取组件作为 LJParser 的一部分，采用灵活的开发接口，可以方便地融入到用户的业务系统中，可以支持各种操作系统，各类调用语言。

3.1 关键词组件函数流程图



3.2 各个函数详细说明

3.2.1 初始化函数

① bool KeyExtract_Init(const char * sDataPath=0,int encode=GBK_CODE,const char*sLicenceCode=0);

功能：为关键词程序准备必要的数据环境；

sDataPath：字典路径；

Encode：编码格式；

sLicenceCode：授权码；

3.2.2 组件系列处理函数

① const char * KeyExtract_GetKeyWords(const char *sLine,int nMaxKeyLimit=50,bool bWeightOut=false);

功能：从字符串中分析关键词，成功返回 true，失败返回 false；

sLine：待处理文本头指针；

nMaxKeyLimit：关键词最大数目，最大不超过 50，默认为 50；

bWeightOut：关键词的输出权重，默认为 0；

② const char * KeyExtract_GetFileKeyWords(const char *sFilename,int nMaxKeyLimit=50,bool bWeightOut=false);

功能：从文本中分析关键词，成功返回 true，失败返回 false；

sLine：待处理文本头指针；

nMaxKeyLimit：关键词最大数目，最大不超过 50，默认为 50；

bWeightOut：关键词的输出权重，默认为 0；

3.2.3 退出函数

① bool KeyExtract_Exit();

功能：退出，释放资源；进程结束前须调用它释放所占用的内存资源

3.3 关键词组件 Java 调用示例

```
public class KeyExtractorTest {

    static {
        if ( CLibraryKeyExtractor.instance.KeyExtract_Init("", 1, "") )
{
            System.out.println("KeyExtractor 初始化成功");
        } else {
            System.out.println("KeyExtractor 初始化失败");
            System.exit(-1);
        }
    }

    public static void main(String[] args) {
        String content = "，全球旅行必备话中蒙友好合作，谈中国周边外交，论亚洲国家相处之道，强调互尊互信、聚同化异、守望相助、合作共赢，共创中蒙关系发展新时代，共促亚洲稳定繁荣";
        String keyWordsStr =
CLibraryKeyExtractor.instance.KeyExtract_GetKeyWords(content, 10, true);
        System.out.println(keyWordsStr);
        CLibraryKeyExtractor.instance.KeyExtract_Exit();
    }
}
```

3.4 关键词组件 C 调用示例

```
if(!KeyExtract_Init(GBK_CODE))//数据在上一层目录下，默认为 GBK 编码的分词
{printf("ICTCLAS  INIT  FAILED!\n");
system("pause");
return  1;
}
const  char  *pKeys=KeyExtract_GetKeyWords("去年开始，打开百度李毅吧，满屏的帖子大多
    含有“屌丝”二字，一般网友不仅不懂这词什么意思，更难理解这个词为什么会这么火。
    然而从下半年开始，“屌丝”已经覆盖网络各个角落，人人争说屌丝，人人争当屌丝。",
    50,true);
//从文本中分析关键词
```

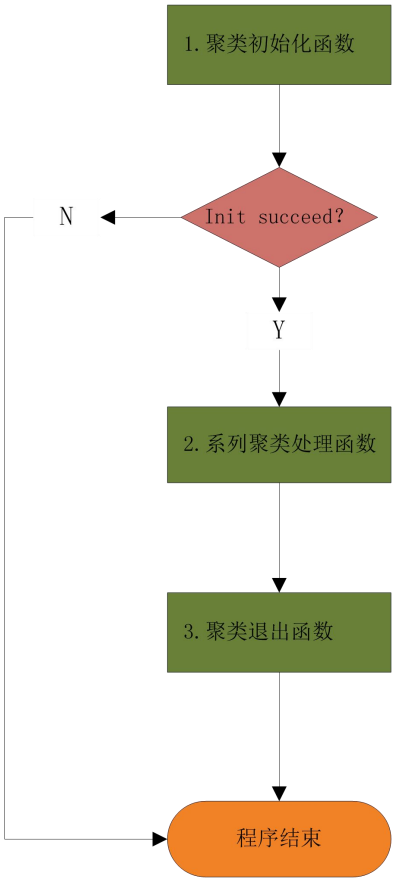


```
printf("Keywords  are:\n%s\n",pKeys);
pKeys=KeyExtract_GetFileKeyWords("D:\\NLPIR\\test\\十八大报告.txt",50,true);
//从文本文件中分析关键词
printf("Keywords  are:\n%s\n",pKeys);
KeyExtract_Exit();
```

四、聚类组件(Cluster) 简要流程图及函数说明文档

聚类组件是基于相似性算法的自动聚类技术，自动对大量无类别的文档进行归类，把内容相近的文档归为一类，并自动为该类生成标题和主题词。适用于自动生成热点舆论专题、重大新闻事件追踪、情报的可视化分析等诸多应用。

4.1 聚类组件函数流程图



4.2 各个函数详细说明

4.2.1 初始化函数

① bool CLUS_Init(const char *sDefaultPath,const char *sLicenseCode=0);

功能：文件方式初始化，成功返回 true，失败返回 false；

conf：字典文件所在路径；

sLicenceCode：授权文件，试用期的一周授权为 NULL；

4.2.2 系列分类处理函数

① bool CLUS_SetParameter(int nMaxClus, int nMaxDoc);

功能：设置参数；

参数：nMaxClus 最多输出前多少个类 nMaxDoc 每个类最多输出前多少个文档

返回：true - 成功；false - 失败

备注：在进程中此函数必须在其他函数之前调用；如果不调用，参数默认均为 2000；

参数越大，系统占用内存越大，处理速度越慢，类和类内的文档均已按照重要性和及时性排序 const char* classifier_detail(const char *classname);

功能：对于当前文档，输入类名，取得结果明细；

classname:结果类名；

返回值：结果明细 例如:

RULE3:

SUBRULE1: 内幕 1

SUBRULE2: 股市 1 基金 3 股票 8

SUBRULE3: 书摘 2

② bool CLUS_AddContent(const char *sText, const char* sSignature);

功能：追加内存内容；

参数：sText - [IN] 正文内容（以'\0'结束的字符串）

sSignature - [IN] 唯一标识符（以'\0'结束的字符串）

返回：true - 成功；false - 失败

备注：在进程中此函数可以在打印结果之前执行多次

③ bool CLUS_AddFile(const char *sFileName, const char* sSignature);

功能：追加文件内容；

参数：sFileName - [IN] 文件全路径名称（以'\0'结束的字符串）

sSignature - [IN] 唯一标识符（以'\0'结束的字符串）

返回：true - 成功；false - 失败

备注：在进程中此函数可以在打印结果之前执行多次

④ bool CLUS_GetLatestResult(const char* sXmlFileName);

功能：打印结果；

参数：sXmlFileName - [OUT]聚类内容输出到 XML 文件中

返回：true - 成功；false - 失败

⑤ void CLUS_CleanData();

功能：清空历史数据；

⑥ const char* CLUS_GetLastErrMsg();

功能：获取错误信息；

4.2.3 分类退出函数

① void CLUS_Exit();

功能：退出，释放资源；进程结束前须调用它释放所占用的内存资源

4. 3 聚类组件 Java 调用示例

```
public class ClusterTest
{
    /**
     * 根据文本文件聚类
     */
    @Test
    public void testCluster0(){
        System.out.println("初始化开始....");
        boolean flag = ClibraryCluster.Instance.CLUS_Init("", "", 1);
        if (flag) {
            System.out.println("初始化成功....");
        } else {
            System.out.println("初始化失败....");
            System.out.println(ClibraryCluster.Instance.CLUS_GetLastErrMsg());
            System.exit(1);
        }

        ClibraryCluster.Instance.CLUS_SetParameter(500, 200);

        /*
         * 读取项目根目录下名为 test 的文件夹内的所有文档
         */
```

```
        File fileDir = new File("test");
        if ( fileDir.isDirectory() ) {
            File[] filesArray = fileDir.listFiles();
            for ( File file : filesArray ) {
                String fileName = file.getName();
                String filePath = file.getAbsolutePath();
                System.out.println("fileName: " + fileName);
                System.out.println("filePath: " + filePath);
                System.out.println("===================================");
                ClibraryCluster.Instance.CLUS_AddFile(filePath, fileName);
            }
        }

        ClibraryCluster.Instance.CLUS_GetLatestResult("d:/ClusterResultByFile.xml");
        ClibraryCluster.Instance.CLUS_Exit();
    }
    /**
     * 根据文本内容聚类
     */
    @Test
    public void testCluster1(){
        System.out.println("初始化开始....");
        boolean flag = ClibraryCluster.Instance.CLUS_Init("", "", 1);
        if (flag) {
            System.out.println("初始化成功....");
        } else {
            System.out.println("初始化失败....");
            System.out.println(ClibraryCluster.Instance.CLUS_GetLastErrMsg());
            System.exit(1);
        }
        ClibraryCluster.Instance.CLUS_SetParameter(1, 2);
        for ( int i = 0; i < 7; i++ ) {
            String content = "周公恐惧流言日，王莽谦恭下士时。假使当年身便死，一生
真伪有谁知。";
            ClibraryCluster.Instance.CLUS_AddContent(content, "content" + i);
        }

        ClibraryCluster.Instance.CLUS_GetLatestResult("d:/ClusterResultByContent.xml");
        ClibraryCluster.Instance.CLUS_Exit();
    }
}
```


4.4 聚类组件 C 调用示例

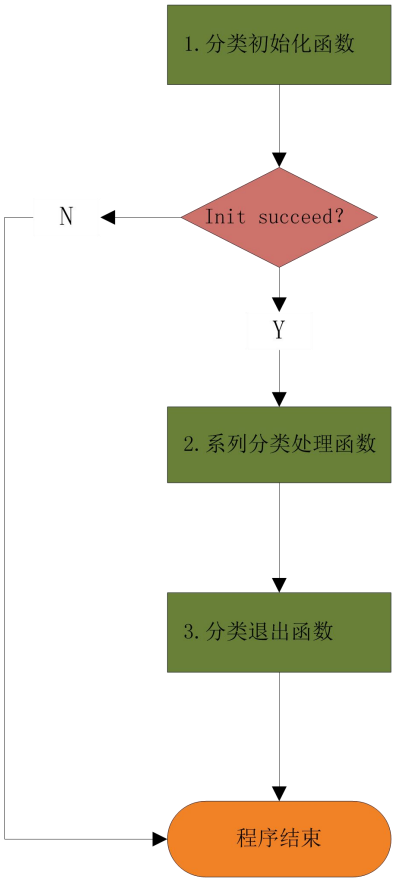
```
if(!CLUS_Init(".\\Data\\Cluster"))
    {
        //获得错误信息
        const char* sMsg = CLUS_GetLastErrMsg();
        printf("分类模块初始化失败: %s\n",sMsg);
        system("pause");
        return -1;
    }
    CLUS_CleanData();
    CLUS_SetParameter(3, 2);
    char sText[100] = "中华人民共和国正式成立的时间是 1949.10.1，代表人名正式当家作主的日子到了";
    CLUS_AddFile("旅游业已成为西部地区的重要产业.txt", "旅游业");
    string sResult;
    //获得聚类处理条件
    CLUS_GetLatestResult("Result.xml");
    gfn_bReadFile("Result.xml", sResult);
    //printf("sResult : %s\n",sResult);
    gfn_vReplaceSubstr(sResult, "\n", "\r\n");
    printf("sResult : %s \n",sResult.c_str());
    CLUS_Exit();
    system("pause");
```

五、分类组件(Classifier) 简要流程图及函数说明文档

分类组件能够根据文献内容进行类别的划分，可以用于新闻分类、简历分类、邮件分类、办公文档分类、区域分类等诸多应用。

它采用基于规则的文本分类过滤以及训练分类两种种方式，并支持两种方式的混合分类。能够进行多级分类，能够进行中英文分类和中英文的混合分类。用户可以灵活、方便的更换模板，来实现对不同的主题的分类过滤。

5.1 分类组件函数流程图



5.2 各个函数详细说明

5.2.1 初始化函数

② bool classifier_init(const char *conf="rulelist.xml",const char*sLicenceCode=NULL);

功能：文件方式初始化，成功返回 true，失败返回 false；

conf：字典文件所在路径；

sLicenceCode：授权文件，试用期的一周授权为 NULL；

5.2.2 系列分类处理函数

⑦ const char* classifier_exec(stDoc& d, int iType=0);

功能：对输入文章结构进行分类；

d：文章结构指针；

iType：iType=0: 输出类名，各类之间用\t 隔开 内容格式举例：“要闻敏感诉讼”；

iType=1: 输出类名和置信度，各类之间用\t 隔开，类名和权重用“ ”隔开

内容格式举例：“要闻 1.00 敏感 0.95 诉讼 0.82”；

iType 默认值为 0；

⑧ const char* classifier_detail(const char *classname);

功能：对于当前文档，输入类名，取得结果明细；

classname:结果类名；

返回值：结果明细 例如:

RULE3:

SUBRULE1: 内幕 1

SUBRULE2: 股市 1 基金 3 股票 8

SUBRULE3: 书摘 2

5.2.3 分类退出函数

① void classifier_exit();

功能：退出，释放资源；进程结束前须调用它释放所占用的内存资源

5. 3 分类组件 Java 调用示例

```
public class ClassifierTest {
  @Test
    public   void Star1() {
        try {
```



```
//                  System.out.println(ClassifierLibrary.Instance_Clussifier);
                  //在项目目录下寻找 classifier.user 文件，自动到 Data 目录下寻找其他文件
                    boolean flag = ClassifierLibrary.Instance.classifier_init("dict/rulelist.xml",0);
                    if(!flag)
                    {
                        System.out.println("初始化失败！\n");
                        return;
                    }
                    System.out.print("初始化状态："+(flag == true?"成功\n\n":"失败\n\n"));
                    String title = "开心";
                    String content = "回来的第一感觉是，移动又要面对中国移动式网络了";
                    String res = ClassifierLibrary.Instance.classifier_exec(title, content, 1);
                    System.out.println("输出结果是:"+res+"\n\n");
            } catch (Exception ex)
            {
                System.err.println(ex);
            }
        }

        }
```

5.4 分类组件 C 调用示例

```
if(!classifier_init("rulelist.xml")) {
        printf("classifier_init failed!\n");
        return 1;
    }
    stDoc d;
    d.sTitle = "天翼定制手机天语 E600";
    d.sContent = "全球旅行必备:天翼定制手机天语 E600 新浪 2011-9-26 15:53 手机——这项人们使用率最高的电子产品,其更新换代速度更是快得无法想象。那么对于我们消费者而言,应当如何选择呢？显然,频繁的换机是非常不划算的,更会增加生活开支,平白增添生活负担。因此,我们在购机之初就应当选择一款满足自身需求的手机。...";
    d.sAuthor = "飞香";
    d.sBoard = "69";
    d.sDatatype = "论坛";
    string sResult = classifier_exec(d);
    printf("\n 过滤结果： %s\n\n", sResult.c_str());
    vector<string> vsItems;
    gfn_bSplit(sResult.c_str(), vsItems, "\t");
    for(int i=0; i<vsItems.size(); i++) {                    //保存分类结果到内存
```



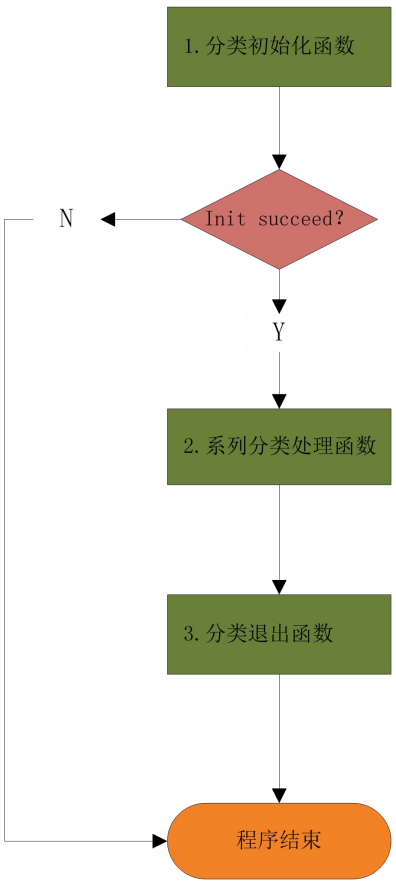
```
        string sDetail = classifier_detail(vsItems[i].c_str());
        printf("过滤理由： %s\n", sDetail.c_str());
    }
    classifier_exit();
    printf("Please input enter to quit...");
    getchar();
```

六、训练分类组件(DeepClassifier) 简要流程图及函数说明文档

分类组件能够根据文献内容进行类别的划分，可以用于新闻分类、简历分类、邮件分类、办公文档分类、区域分类等诸多应用。

训练分类是指基于已分类好的语料自动将新的语料进行归类，灵玖采用基于规则的文本分类以及训练分类两种种方式，并支持两种方式的混合分类，能够进行多级分类，能够进行中英文分类和中英文的混合分类。用户可以灵活、方便的更换模板，来实现对不同的主题的分类过滤。

6.1 训练分类组件函数流程图



6.2 各个函数详细说明

6.2.1 初始化函数

③ boolean DC_Init(constchar*sDataPath,int encode,int nFeatureCount,const char*sLicenceCode);

功能：文件方式初始化，成功返回 true，失败返回 false；

备注：初始化函数运行一次

参数：sDataPath - Data 文件夹的路径，如果为空字符串会从项目的根目录下寻找

Encode - 训练文本的编码， 0 为 gbk，1 为 utf-8

nFeatureCount -

sLicenceCode - 授权码，置为空字符串就可以了

6.2.2 系列分类处理函数

① bool DC_AddTrain(const char *sClassName,const char *sText)

功能：添加训练文本

参数：sClassName - 分类的类名（例如：人物、地理、历史、政治）

sText - 文本内容

② bool DC_AddTrainFile(const char *sClassName,const char *sFilename);

功能：添加训练文本文件

参数：sClassName - 分类的类名（例如：人物、地理、历史、政治）

SFilename - 文件的路径名

返回：true-成功，false-失败

③ bool DC_Train();

功能：DeepClassifier Training on given text in Memory

After training, the training result will stored.

Then the classifier can load it with DC_LoadTrainResult at any time(offline or online).

返回：true-成功，false-失败

④ bool DC_LoadTrainResult();

功能：DeepClassifier Load already training data

备注：DC_Train（）方法调用成功后调用该方法

返回：true-成功，false-失败

⑤ const char * DC_Classify(const char *sText);

功能：针对文本内容进行分类判断。

参数：sText - 文本内容

返回值：文本内容所属的类别

⑥ const char * DC_ClassifyFile(const char *sFilename);

功能：针对文本文件进行分类判断。

参数：sFilename - 文本文件的路径名

返回值：文本文件所属的类别。

⑦ const char * DC_GetLastErrorMsg();

功能：返回最新一次的错误信息

返回：最新一次的错误信息

6.2.3 分类退出函数

① void DC_Exit();

功能：退出，释放资源；进程结束前须调用它释放所占用的内存资源

6.3 训练分类组件 Java 调用示例

```
public class DeepClassifierTest {
    public static void main(String[] args) throws Exception {
        ////////////////////////////////////
        //
        //训练过程
        //
        ////////////////////////////////////
        //1、训练过程--初始化
        boolean flag = DeepClassifierLibrary.Instance.DC_Init("", 0, 800, "");
        if (flag) {
            System.out.println("deepClassifier 初始化成功");
        } else {
            System.out.println("deepClassifier 初始化失败：" + DeepClassifierLibrary.Instan
ce.DC_GetLastErrorMsg());
            System.exit(1);
        }
        //2、训练过程--遍历训练分类文本的文件夹，添加所有的训练分类文本
        ArrayList list = FileOperateUtils.getAllFilesPath(new File("训练分类用文本"));
        for (int i = 0; i < list.size(); i++) {
            File f = new File(list.get(i).toString());
```



```
                String className = f.getParent();
                className = className
                        .substring(className.lastIndexOf("\\") + 1);
                //将训练分类文本加载到内存中
                DeepClassifierLibrary.Instance.DC_AddTrain(
                        className, FileOperateUtils.getFileContent(list.get(i).toString(), "gbk
"));
        }
        //3、训练过程--开始训练
        DeepClassifierLibrary.Instance.DC_Train();
        //4、训练过程--训练结束，退出
        DeepClassifierLibrary.Instance.DC_Exit();
        ////////////////////////////////////
        //
        //分类过程
        //
        ////////////////////////////////////
        //1、分类过程--初始化
        if (DeepClassifierLibrary.Instance.DC_Init("", 0, 800, "")) {
            System.out.println("deepClassifier 初始化成功");
        } else {
            System.out.println("deepClassifier 初始化失败：" + DeepClassifierLibrary.Instan
ce.DC_GetLastErrorMsg());
            System.exit(1);
        }
        //2、分类过程--加载训练结果
        DeepClassifierLibrary.Instance.DC_LoadTrainResult();
        //3、分类过程--读取待分类的文本
        String content = FileOperateUtils.getFileContent("abc.txt", "gbk");
        //4、分类过程--输出分类结果
        System.out.println("分类结果：" + DeepClassifierLibrary.Instance.DC_Classify(conte
nt));
        //5、分类过程--退出
        DeepClassifierLibrary.Instance.DC_Exit();
    }

}
```

6.4 训练分类组件 C 调用示例

```
const double ratio = 0.8;
```



```
//const string working_path = "D:/NLPIR/Data/DeepClassifier/";
if (!DC_Init(working_path.c_str(),GBK_CODE))
{
printf("Init Failed! Reason is %s\n",DC_GetLastErrorMsg());
return ;
}
std::map < string, vector<string> > class_map;
vector<string> dirs;
    get_dirs(path, dirs);
    int   nFileCount=0;
    for (vector<string>::iterator p = dirs.begin(); p != dirs.end(); ++p)
{
        string spath = path+*p+"/";
        cout << "process : "<< spath << endl;
        vector<string> files;
        get_files(spath, files);
        nFileCount+=files.size();
        class_map[*p] = files;
    }
    dirs.clear();
    printf("Start to training!\n");
    int count = 0;
    nFileCount*=ratio;
    for (std::map< string, vector<string> >::iterator p = class_map.begin(); p != class_map.
end(); ++p)
    {const int maxindex = p->second.size() * ratio;
        for (int i = 0; i < maxindex; ++i)
{
            string spath = path + p->first + "/" + p->second[i];
            //cout << "process : "<< spath<< "i"<< i <<"maxindex"<< maxindex <<endl;
            DC_AddTrainFile(p->first.c_str(), spath.c_str());
            ++count;
            printf("Training %d%%(%d/%d)\r",count*100/nFileCount,count,nFileCount);
        }
    }
    DC_AddTrainComplete();
    printf("DC_AddTrainComplete Completed!\n");
    DC_Train();
    printf("DC_Train Completed!\n");
    DC_LoadTrainResult();
    printf("DC_LoadTrainResult Completed!\n");
    //????????????????????????????precision??
    //unordered_map<string, int> c_info1;
    map<string, int> c_info1;
```



```
    //??????????????????
    map<string, int> c_info2;
    map<string, double> recall;
    int count_all_right = 0;
    int count_all = 0;
    for (std::map< string, vector<string> >::iterator p = class_map.begin(); p != class_map.
end(); ++p)
{
        c_info2[p->first] = 0;
    }
    for (std::map< string, vector<string> >::iterator p = class_map.begin(); p != class_map.
end(); ++p) {
        int count_right = 0;
        int count = 0;
        const int min = p->second.size() * ratio;
        for (int i = min; i < p->second.size(); ++i) {
            //for (int i = 0; i < min; ++i) {
            string spath = path + p->first + "/" + p->second[i];
            string sc = DC_ClassifyFile(spath.c_str());
            ++count;
            ++count_all;
            c_info1[sc]++;
            //cout << p->first << endl;
            if(p->first == sc) {
                ++count_right;
                ++count_all_right;
            }
        }
        c_info2[p->first] = count_right;
        cout << endl;
        cout << p->first <<  " ?? right number : " << count_right << endl;
        //cout << "Recall Rate : " << count_right / (double)vf.size() << endl;
        recall[p->first] = count_right / (double)count;
    }
    int all = 0;
    cout << c_info1.size() << endl;
    //ofstream outfile(, ios::app);
    std::string sFile=working_path+"result.txt";
    FILE *outfile=fopen(sFile.c_str(),"ab");
    if (outfile==0) {
        printf("open outifle %s error \n",sFile.c_str());
        return;
    }
//    system_clock::time_point today = system_clock::now();
```

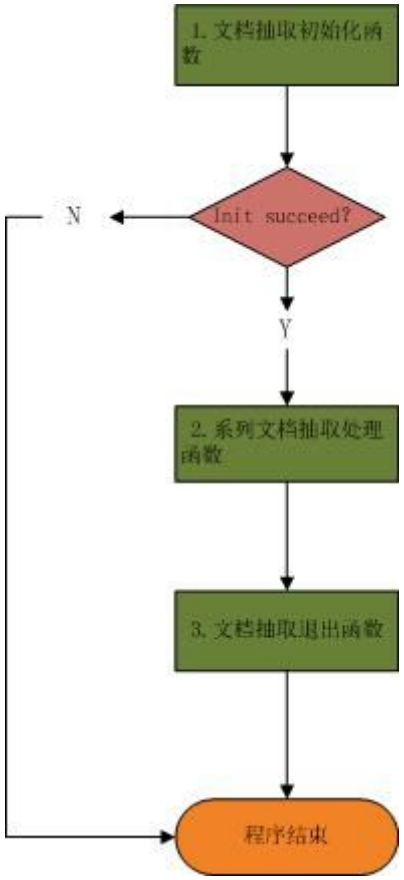


```
//	std::time_t tt = system_clock::to_time_t (today);
	time_t tt;
	time(&tt);
	string sTime=string(ctime(&tt));
	fprintf(outfile,"%s\nfeatures : \t%d\n",sTime.c_str(), num);
	fprintf(outfile,"class name\ttraining set\ttesting set\tprecision\trecall rate\tF value\n");
//	outfile << "class name\t"<<"training set\t"<<"testing set\t"<<"precision\t"<<"recall rate\t
"<<"F value\n";
	for (std::map<string, int>::iterator p = c_info1.begin(); p != c_info1.end(); ++p)
	{
		all += p->second;
		const double precision = c_info2[p->first] / (double)p->second;
		//fprintf(outfile,"class %s\n",p->first.c_str());
		cout << "class " << p->first << " : \n";
		//fprintf(outfile,"precision : %.2f\n",precision);
		cout << "precision : " << precision << endl;
		//fprintf(outfile,"recall rate : %.2f\n",recall[p->first]);
cout << "recall rate : " << recall[p->first] << endl;cout << "F : " << precision * recall[p
->first] * 2 /(recall[p->first] + precision) << endl;
	fprintf(outfile,"%s\t%d\t%d\t%lf\t%lf\t%lf\n",
			p->first.c_str(),
			int(class_map[p->first].size() * ratio),
			(int) (class_map[p->first].size() * (1.0-ratio)),
			precision,
			recall[p->first],
			precision * recall[p->first] * 2 /(recall[p->first] + precision)
			);
	/*	outfile << p->first << "\t" << int(class_map[p->first].size() * ratio) << "\t"
			<< (int) (class_map[p->first].size() * (1.0-ratio)) << "\t" << precision << "\t"
 << recall[p->first] << "\t"
			<< precision * recall[p->first] * 2 /(recall[p->first] + precision) << endl;*/
	}
	const double drecall = count_all_right / (double)count_all;
	//outfile << "total :\t" << drecall << endl << endl;
	fprintf(outfile,"total :\t%lf\n",
		Drecall
		);
	//const double precision = count_all_right / (double)all;
	//cout << "total recall rate : " << drecall << endl;
	//cout << "total precision : " << precision << endl;
	//cout << "total F : " << 2 * precision * drecall /(precision + drecall) << endl;
	//cout << "total accuracy :\t" << drecall << endl;
	fclose(outfile);
	DC_Exit();
```


七、文档抽取组件(DocExtractor) 简要流程图及函数说明文档

文档抽取（DocExtractor）组件是指自动识别包含在自然语言文本中的实体之间的预定义关系。所谓实体是指文本中包含的特定事实信息,如人物、组织机构、地理位置等。文档抽取在数据结构化、信息检索和自动应答系统等领域有着重要的研究意义。美国国家标准技术研究院(NIST)在 2008 年组织的自动内容抽取(ACE,AutomaticContentExtraction)评测中定义了 7 种实体关系类型和 18 种子类型。

7.1 文档抽取组件函数流程图：



7.2 各个函数详细说明

7.2.1 初始化函数

① int DE_Init(const char *sPath=0,int nEncoding=GBK_CODE,const char

*sLicenseCode=0);

功能：初始化

返回：1 - 成功；0 - 失败

备注：在进程中此函数必须在其他函数之前调用（只需执行一次）

参数：sPath - Data 文件夹的路径，为空字符串时默认从工程根目录下开始寻找

nEncoding - 编码格式，具体如下：

0：GBK；1：UTF8；2：BIG5；3：GBK（里面包含繁体字）

sLicenseCode - 授权码,写成空字符串就可以

7.2.2 系列文档抽取处理函数

① DOC_PARSER_HANDE DE_ParseDocE(const char *sText, const char *sUserDefPos, bool bSummaryNeeded, unsigned int nFuncRequired)

功能：处理文档，生成 handle 值，用于后续的文档处理操作

备注：在进程中此函数可以执行多次

参数： sText - 文档内容

sUserDefPos - 用户自定义的词性标记，最多三种（人名、地名、机构名、媒体等内置，无需设置），不同词类之间采用#分割，如 "gms#gjtgj#g"

bSummaryNeeded=true-[IN]:是否需要计算摘要，true：需要；false：不需要

nFuncRequired - 需要抽取的类型，所有的类型如下

PERSON_REQUIRED	0x0001
LOCATION_REQUIRED	0x0002
ORGANIZATION_REQUIRED	0x0004
KEYWORD_REQUIRED	0x0008
AUTHOR_REQUIRED	0x0010
MEDIA_REQUIRED	0x0100
COUNTRY_REQUIRED	0x0200
PROVINCE_REQUIRED	0x0400
ABSTRACT_REQUIRED	0x0800

SENTIWORD_REQUIRED 0x1000

SENTIMENT_REQUIRED 0x2000

USER_REQUIRED 0x4000

HTML_REMOVER_REQUIRED 0x8000// 是否需要去除网页标签的功能选项

ALL_REQUIRED 0xffff

② const char *DE_GetResult(DOC_PARSER_HANDLE handle,int nDocExtractType);

功能：获得抽取结果

备注：在进程中此函数可以执行多次

参数：handle - DE_ParseDocE 获得的值

nDocExtractType - 抽取的类型，所有类型如下：

DOC_EXTRACT_TYPE_PERSON 0//输出的人名

DOC_EXTRACT_TYPE_LOCATION 1//输出的地名

DOC_EXTRACT_TYPE_ORGANIZATION 2//输出的机构名

DOC_EXTRACT_TYPE_KEYWORD 3//输出的关键词

DOC_EXTRACT_TYPE_AUTHOR 4//输出的文章作者

DOC_EXTRACT_TYPE_MEDIA 5//输出的媒体

DOC_EXTRACT_TYPE_COUNTRY 6//输出的文章对应的所在国别

DOC_EXTRACT_TYPE_PROVINCE 7//输出的文章对应的所在省份

DOC_EXTRACT_TYPE_ABSTRACT 8//输出文章的摘要

DOC_EXTRACT_TYPE_POSITIVE 9//输出文章的正面情感词

DOC_EXTRACT_TYPE_NEGATIVE 10//输出文章的副面情感词

DOC_EXTRACT_TYPE_TEXT 11//输出文章去除网页等标签后的正文

DOC_EXTRACT_TYPE_USER 12//用户自定义的词类，第一个自定义词

③ int DE_GetSentimentScore(DOC_PARSER_HANDLE handle);

功能：获得情感值

返回：情感值

参数：Handle - DE_ParseDocE 获得的值

④ void DE_ReleaseHandle(DOC_PARSER_HANDLE handle);

功能：释放 handle

备注：文档抽取结束后应调用该方法，释放 handle

参数：Handle - DE_ParseDocE 获得的值

⑤ unsigned int DE_ImportSentimentDict(const char *sFilename);

功能：导入用户自定义的情感词表，每行一个词，空格后加上正负权重，如：语焉不详 -2;

备注：如果导入的情感词属于新词，会在用户词典中导入，否则情感识别自动跳跃

参数：sFilename - 情感词典的路径。（请使用全路径的形式）

⑥ unsigned int DE_ImportUserDict(const char *sFilename,bool bOverwite=true);

功能：导入用户词典

备注：系统会自动处理重复词的问题

参数：sFilename - 自定义用户词典的路径。（请使用全路径的形式）

bOverwite - 是否覆盖上次添加的自定义用户词典，true：覆盖；false：不覆盖

⑦ unsigned int DE_ImportKeyBlackList(const char *sFilename);

功能：导入关键词黑名单

参数：sFilename - 黑名单词典的路径。（请使用全路径的形式）

⑧ int DE_ComputeSentimentDoc(const char *sText);

功能：计算文档情感值

返回：情感值

参数：文档内容

⑨ const char* DE_GetLastErrMsg();

功能：获得最近一次的错误消息

返回：最近一次错误消息

7.2.3 退出函数

① void DE_Exit();

功能：退出,释放系统资源

7.3 文档抽取组件 Java 调用示例

```
public class DocExtractTest {
    public static Logger logger = Logger.getLogger(DocExtractTest.class);

    private static final int DOC_EXTRACT_TYPE_PERSON = 0;// 输出的人名
    private static final int DOC_EXTRACT_TYPE_LOCATION = 1;// 输出的地名
    private static final int DOC_EXTRACT_TYPE_ORGANIZATION = 2;// 输出的机构名
    private static final int DOC_EXTRACT_TYPE_KEYWORD = 3;// 输出的关键词
    private static final int DOC_EXTRACT_TYPE_AUTHOR = 4;// 输出的文章作者
    private static final int DOC_EXTRACT_TYPE_MEDIA = 5;// 输出的媒体
    private static final int DOC_EXTRACT_TYPE_COUNTRY = 6;// 输出的文章对应的所在国别
    private static final int DOC_EXTRACT_TYPE_PROVINCE = 7;// 输出的文章对应的所在省份
    private static final int DOC_EXTRACT_TYPE_ABSTRACT = 8;// 输出文章的摘要
    private static final int DOC_EXTRACT_TYPE_POSITIVE = 9;// 输出文章的正面情感词为
    private static final int DOC_EXTRACT_TYPE_NEGATIVE = 10;// 输出文章的副面情感词
    private static final int DOC_EXTRACT_TYPE_DEL_HTML = 11;// 输出文章去除网页等标签后的正文
    private static final int DOC_EXTRACT_TYPE_USER_DEFINED1 = 12;//用户自定义词 1
    // .....

    private static final int PERSON_REQUIRED = 0x0001;
    private static final int LOCATION_REQUIRED = 0x0002;
    private static final int ORGANIZATION_REQUIRED = 0x0004;
    private static final int KEYWORD_REQUIRED = 0x0008;
    private static final int AUTHOR_REQUIRED = 0x0010;
    private static final int MEDIA_REQUIRED = 0x0100;
    private static final int COUNTRY_REQUIRED = 0x0200;
    private static final int PROVINCE_REQUIRED = 0x0400;
    private static final int ABSTRACT_REQUIRED = 0x0800;
    private static final int TRANS_REQUIRED = 0x1000;
    private static final int FOOD_REQUIRED = 0x2000;
    private static final int APPS_REQUIRED = 0x4000;
    private static final int SENTIMENT_REQUIRED = 0x8000;
    private static final int ALL_REQUIRED = 0xffff;

    @Test
    public void testImportUserDict() {
        if ( DocExtractLibray.Instance.DE_Init("", 1, "") == 0 ) {
            System.out.println("DocExtractor 初始化失败：" +
```

```
DocExtractLibray.Instance.DE_GetLastErrMsg());
                System.exit(1);
            }
            System.out.println("DocExtractor 初始化成功");
            System.out.println("成功导入的自定义词个数：" +
DocExtractLibray.Instance.DE_ImportUserDict("dict/userDict.txt"));
        }

        @Test
        public void testExtratContent() {
            for (int i = 0; i < 1; i++) {
                if ( DocExtractLibray.Instance.DE_Init("", 1, "") == 0 ) {
                    System.out.println("初始化失败：" +
DocExtractLibray.Instance.DE_GetLastErrMsg());
                    System.exit(1);
                }
                System.out.println("初始化成功");
            String content = "&lt;p&gt;　　本报北京 10 月 28 日电应老挝人民革命党和越南共
产党邀请，中共中央政治局委员、中央书记处书记、中宣部部长刘奇葆将率中共代表团于
10 月 31 日至 11 月 5 日赴老挝、越南出席主题为“建设社会主义法治国家的经验”的第三次
中老两党理论研讨会和第十次中越两党理论研讨会并访问老挝。 &lt;/p&gt;&lt;p&gt;
&lt;br /&gt;&lt;/p&gt;　　《 人民日报 》（ 2014 年 10 月 29 日 06 版）";
                int score=DocExtractLibray.Instance.DE_ComputeSentimentDoc(content);
                System.out.println("--->score--->"+score);
                NativeLong handle = DocExtractLibray.Instance.DE_ParseDocE(content,
"mgc#ngd",
                        true, ALL_REQUIRED);
                System.out.println("抽取的人名为-->"
                        + DocExtractLibray.Instance.DE_GetResult(handle, 0));
                System.out.println("抽取的地名为-->"
                        + DocExtractLibray.Instance.DE_GetResult(handle, 1));
                System.out.println("抽取的机构名为-->"
                        + DocExtractLibray.Instance.DE_GetResult(handle, 2));
                System.out.println("抽取的关键词为-->"
                        + DocExtractLibray.Instance.DE_GetResult(handle, 3));
                System.out.println("抽取的文章作者为-->"
                        + DocExtractLibray.Instance.DE_GetResult(handle, 4));
                System.out.println("抽取的媒体为-->"
                        + DocExtractLibray.Instance.DE_GetResult(handle, 5));
                System.out.println("抽取的文章对应的所在国别为-->"
                        + DocExtractLibray.Instance.DE_GetResult(handle, 6));
                System.out.println("抽取的文章对应的所在省份为-->"
                        + DocExtractLibray.Instance.DE_GetResult(handle, 7));
                System.out.println("抽取的文章摘要为-->"
```



```
                    + DocExtractLibray.Instance.DE_GetResult(handle, 8));
            System.out.println("输出文章的正面情感词为-->"
                    + DocExtractLibray.Instance.DE_GetResult(handle, 9));
            System.out.println("输出文章的副面情感词-->"
                    + DocExtractLibray.Instance.DE_GetResult(handle, 10));
            System.out.println("输出文章原文-->" + content);
            System.out.println("输出文章去除网页等标签后的正文-->"
                    + DocExtractLibray.Instance.DE_GetResult(handle, 11));
            System.out.println("去除空格:" +
DocExtractLibray.Instance.DE_GetResult(handle, 11).replaceAll("[    *| *| *|//s*]*", ""));

            System.out.println("自定义词(mgc)-->"
                    + DocExtractLibray.Instance.DE_GetResult(handle, 12));
            System.out.println("情感值----->" +
DocExtractLibray.Instance.DE_GetSentimentScore(handle));
            DocExtractLibray.Instance.DE_ReleaseHandle(handle);

            System.out.println("是否安全退出-->"+DocExtractLibray.Instance.DE_Exit());
        }
    }

}
```

7.4 文档抽取组件 C 调用示例

```
if (argc != 2){
        printf("usage: %s <CorpusDir>\n",argv[0]);
        return 1;
    }
    if(!DE_Init("",GBK_CODE)) {
        printf("%s\n", DE_GetLastErrMsg());
        return 1;
    }
    // 扫描
    int nRealCount;
    vector<string> vsFileName;
```

```
fn_vScanFiles(argv[1], vsFileName,"txt");
//fn_vScanFiles(std::string& sFolderName, std::vector<std::string>& vsFileName,tstring& sFilter)
string sOutput = "";
// 加载
printf("分析文章（Total %d）...\n",vsFileName.size());
int i;
clock_t tStart,tEnd;
long lTime,lTotalTime=0,lTotalSize=0;
float fTime;
_tDocExtractResult result;
for(i=0; i<vsFileName.size(); i++) {
    string sContent;
    gfn_bReadFile(vsFileName[i].c_str(), sContent);
    tStart=clock();
    /*printf("%s\n",NLPIR_ParagraphProcess(sContent.c_str()));
    int nSent = DE_ComputeSentimentDoc(sContent.c_str());
    printf("SENTIMENT:%d\n",nSent);*/
DE_ParseDoc(sContent.c_str(),result,'\0');//,LOCATION_REQUIRED|ORGANIZATION_REQUIRED|PERSON_REQUIRED
    tEnd=clock();
    lTime=tEnd-tStart;
    lTotalTime+=lTime;
    lTotalSize+=sContent.size();
    printf("-------------------------------------\n 文件:%s\n", vsFileName[i].c_str());
    printf("Person list:%s\n",result.entity_list[DOC_EXTRACT_TYPE_PERSON]);
    printf("Loc list:%s\n",result.entity_list[DOC_EXTRACT_TYPE_LOCATION]);
    printf("Org list:%s\n",result.entity_list[DOC_EXTRACT_TYPE_ORGANIZATION]);
    printf("Abstract list:%s\n",result.entity_list[DOC_EXTRACT_TYPE_ABSTRACT]);
    printf("Keyword list:%s\n",result.entity_list[DOC_EXTRACT_TYPE_KEYWORD]);
```



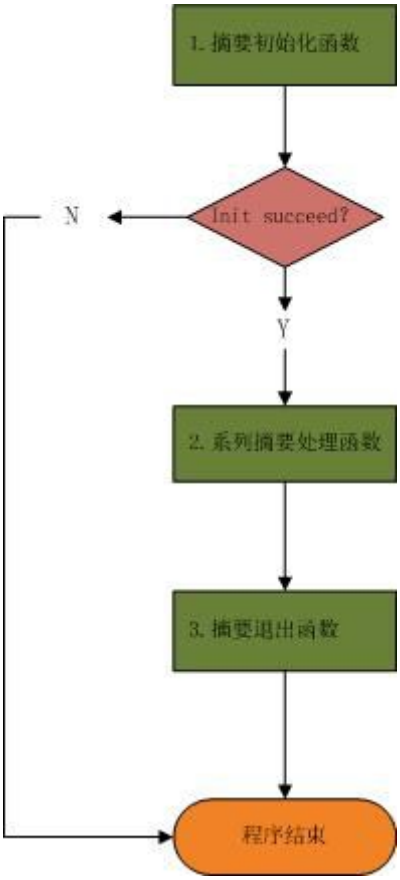
```
            printf("Media list:%s\n",result.entity_list[DOC_EXTRACT_TYPE_MEDIA]);
            //sOutput += pResult;
            //sOutput += "\n";
            fTime=(float)lTime/(float)CLOCKS_PER_SEC;//Time cost
            printf("Size=%d                                                    Bytes,time=%.2f
s,speed=%.2fKB/s\n",sContent.size(),fTime,sContent.size()/fTime/1000);
            //getchar();
        }
        // 退出
        DE_Exit();
        //NLPIR_Exit();
        fTime=(float)lTotalTime/(float)CLOCKS_PER_SEC;//Time cost
        printf("Total          Size=%d          bytes,total          time=%.2fs,average
speed=%.2fKB/s\n",lTotalSize,fTime,lTotalSize/fTime/1000);
```


八、摘要组件(Summary) 简要流程图及函数说明文档

摘要组件能够实现文本内容的精简提炼，从长篇文章中自动提取关键句和关键段落，构成摘要内容，方便用户快速浏览文本内容，提高工作效率。

自动摘要组件不仅可以针对一篇文档生成连贯流程的摘要，还能够将具有相同主题的多篇文档去除冗余、并生成一篇简明扼要的摘要；用户可以自由设定摘要的长度、百分比等参数；处理速度达到每秒钟 20 篇。

8.1 摘要组件函数流程图



8.2 各个函数详细说明

8.2.1 初始化函数

① int DS_Init(const char *sPath=0,int nEncoding=GBK_CODE,const char *sLicenseCode=0);

功能：初始化

备注：在进程中此函数必须在其他函数之前调用（只需执行一次）

返回：1- 成功；0 - 失败

参数： sPath - Data 文件夹的路径，为空字符串时默认从工程根目录下开始寻找

nEncoding - 编码格式，具体如下：

0：GBK；1：UTF8；2：BIG5；3：GBK（里面包含繁体字）

sLicenseCode - 授权码,写成空字符串就可以

8.2.2 系列摘要处理函数

① const char* DS_SingleDoc(const char *sText, float fSumRate=0.00, int iSumLen=250,bool bHtmlTagRemove=false);

功能：生成单文档摘要

返回：摘要字符串；出错返回空串

备注：在进程中此函数可以执行多次

参数： sText - 文档内容

fSumRate -[IN] 文档摘要占原文百分比（范围 0~1），

使用该参数时，把 iSumLen 置为 0

iSumLen -[IN] 用户限定的摘要长度

使用该参数时，把 fSumRate 置为 0

bHtmlTagRemove-[IN] 是否需要对原文进行 Html 标签的去除，

true 表示去除 html 标签；false 表示不去除

② const char* DS_SingleDocE(char *sResult,const char *sText, float fSumRate=0.00, int iSumLen=250,bool bHtmlTagRemove=false);

功能：生成单文档摘要该函数支持多线程，是多线程安全的

返回：摘要字符串；出错返回空串

备注：在进程中此函数可以执行多次，该方法是 c 语言用的，java 不要调用该方法。

参数： sResult -[IN] 摘要内容

sText -[IN] 文档内容

fSumRate -[IN] 文档摘要占原文百分比

iSumLen -[IN] 用户限定的摘要长度

bHtmlTagRemove-[IN] 是否需要对原文进行 Html 标签的去除

③ const char* DS_GetLastErrMsg();

功能：获得最近一次的错误消息

返回：最近一次错误消息

8.2.3 摘要退出函数

② void DS_Exit();

功能：退出，释放资源；进程结束前须调用它释放所占用的内存资源

8.3 摘要组件 Java 调用示例

```
public class SummaryTest {
    static {
        //初始化
        boolean flag = CLibraryDS.Instance.DS_Init("", 1, "0");
        if (flag == false) {//如果初始化失败，就打印出失败原因
            System.out.println(CLibraryDS.Instance.DS_GetLastErrMsg());
        }
    }

    public static void main(String[] args) {
        //文本内容
        long startTime = System.currentTimeMillis();
//          String content = "10 月 27 日，在山东省无棣县劳动就业处培训中心，部分创业人员正在辅导老师的指导下完善自己刚建的淘宝店铺。为激励青年创业的积极性，该县劳动就业处从今年 6 月份开始分期分批免费对初始创业人员进行电子商务创业培训，使其熟练掌握电子商务相关知识。截至目前，超过 3000 人完成技能培训，就业率达到 95%。本报记者徐烨摄《 人民日报 》（ 2014 年 10 月 28 日 10 版）";
        String content = FileOperateUtils.getFileContent("sm1.txt");
        System.out.println("原内容： " + content);
        System.out.println(content.length());
        String summary = CLibraryDS.Instance.DS_SingleDoc(content, 0f, 80, true);
        System.out.println("摘要为： " + summary);
        System.out.println(summary.length());
        long endTime = System.currentTimeMillis();
        System.out.println("共耗时：" + ((endTime - startTime) / 1000) + "s");
    }
```



```
}
```

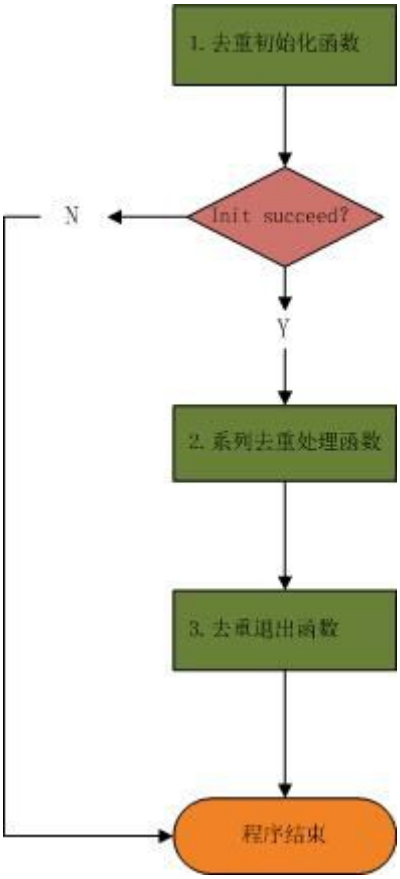
8.4 摘要组件 C 调用示例

```
        /*第一个参数为空表示当前工作路径*/
        if(!DS_Init("",GBK_CODE))
        {
            printf("sErrorMsg : %s \n", DS_GetLastErrMsg());
            return -1;
        }
        vector<string> vsFileName;
        /*遍历 argv[1]文件夹将所有文件名称保存在 vsFileName 中*/
        fn_vScanFiles(argv[1], vsFileName,"txt");
        string sOutput = "";
        FILE* fd ;
        // 加载
        printf("分析文章（Total %d）...\n",vsFileName.size());
        for(int i=0; i<vsFileName.size(); i++)
    {
            string sContent;
            printf("-------->,,name : %s\n",vsFileName[i].c_str());
            /*读取文件内容到 sContent*/
            gfn_bReadFile(vsFileName[i].c_str(), sContent);
            const char* sSummary = DS_SingleDoc(sContent.c_str());
            printf("[1]本篇文章的摘要是 SingleDoc：%s \n", sSummary);
            fd = fopen(vsFileName[i].c_str(),"wb");
            if(fd)
    {
                fwrite(sSummary,sizeof(char),strlen(sSummary),fd);
                fclose(fd);
            }

        // char szSummary[1024] = {0};

        // const char* sRes = DS_SingleDocE(szSummary,sContent.c_str());

        // printf("[2]本篇文章的摘要是 SingleDocE：%s \n, [3]%s\n", sRes,szSummary);

    }

    //const char* DS_SingleDoc(const char *sText, float fSumRate=0.00, int iSumLen=25
0);

    DS_Exit();
```


九、去重组件(RedupRemover) 简要流程图及函数说明文档

去重组件能够快速准确地判断文件集合或数据库中是否存在相同或相似内容的记录，同时找出所有的重复记录。运行环境可以支持 Windows、Linux、FreeBSD 等多种环境，支持普通 PC 机器即可运行；支持 GBK/UTF-8/BIG5。

9.1 去重组件函数流程图



9.2 各个函数详细说明

9.2.1 初始化函数

① bool RR_Init(const char *sHistoryDataFile, const char *sInitDirPath,bool bHistoryLoad, const char *sLicenseCode, int encode);

功能：初始化,并装载已有的查重数据

备注：在进程中此函数必须在其他函数之前调用（只需执行一次）

参数：sHistoryDataFile: 查重历史数据文件的路径，如果没有或者不想使用就置为空字

符串

sInitDirPath: Data 文件夹的路径，为空字符串时默认从工程根目录下开始寻找

bHistoryLoad: 是否加载历史数据，默认为否

sLicenseCode - 授权码,写成空字符串就可以

encode - 文档的编码，具体对照如下：

0：GBK；1：UTF8；2：BIG5；3：GBK（里面包含繁体字）

9.2.2 系列去重处理函数

① int RR_FileProcess(const char *sContent, const char *sID);

功能：处理文档

返回：返回 1 说明该文档是重复的，否则是不重复的

参数：sContent: 文本内容

sID: 能够唯一标识该文本的字符串（比如：可以是文本的标题）

② bool RR_FindRepeat(char *sReturnString, bool bAllOutput = false)

功能：处理发现的重复文档

备注：该方法在 RR_FileProcess 的返回值是 1 时可以调用

参数：sReturnString - 返回结果

bAllOutput - true：返回所有重复的文件；false：只返回当前文件

③ bool RR_Output(const char *sHistoryDataFile = "RepeatFile.txt");

功能：保存重复文件信息

参数：sHistoryDataFile - 保存路径

④ bool RR_SaveHistoryData(const char *DataFilePath = "Data.txt")

功能：保存查重历史数据，可用于下次加载，增量处理

参数：sHistoryDataFile - 查重历史数据文件路径

⑤ const char* RR_GetLastErrMsg();

功能：获得最近一次的错误消息

返回：最近一次错误消息

9.2.3 去重退出函数

① void RR_Exit();

功能：退出，释放资源，进程结束前须调用它释放所占用的内存资源

9.3 去重组件 Java 调用示例

```
public class RedupRemoverTest
{
    @Test
    public void testRedup2() throws UnsupportedEncodingException{
        if(!RedupRemoverLibrary.Instance.RR_Init("","",false,"", 1)){
            System.out.println("初始化失败");
            System.out.println(RedupRemoverLibrary.Instance.RR_GetLastErrMsg());
        }else{
            System.out.println("初始化成功");
            List<text> l = new ArrayList<text>();
            for ( int i = 0; i < 15; i++ ) {
                text t = new text();
                t.setId(i);
                t.setTitl("标题"+i);
                t.setName("name"+i);
                if(i<5){
                    t.setText("中文中文中中文中文中中文中文中中文中文中");
                }else if ( i < 10 ){
                    t.setText("大众新速腾车主不接受召回方案 在美受损车辆国考最热职位竞争比达 1300:1 有 700 余职位零报冯仑隐退江湖还是投身理想城 实惠走进校园免费送福利科技 |[第一线]苍井空做内衣电商：半数买家为男性苹果发布 iOS 8.1 升级指南 iPad 销量连续三个季度下滑丛林偶遇世界最大蜘蛛似幼犬 英一家人养 145 只动物同住时尚 | 抢镜必备“一抹蚊子血” 高圆圆周迅新婚娇美热点 | 听军事家解析时局热点 体育赛事一网打尽关注 | 协力找马航失联客机 昌平苹果甜蜜的选择");
                } else {
                    t.setText("周公恐惧流言日，王莽谦恭下士时");
                }
                System.out.println(t.getText());
                l.add(t);
            }

            byte[] pcFindAll = new byte[65535];
            for(int i=0;i<l.size();i++){
```

```
                    System.out.println("处理到第---->"+(i)+"/"+(l.size()-1));
                    if (RedupRemoverLibrary.Instance.RR_FileProcess(l.get(i).getText(),
l.get(i).getId() + l.get(i).getTitl()) == 1)
{
                        System.out.println("find repead data");
                        RedupRemoverLibrary.Instance.RR_FindRepeat(pcFindAll, true);
                        System.out.println("pcFindAll--->"+new String(pcFindAll));
                    }
                }
                pcFindAll = null;
                RedupRemoverLibrary.Instance.RR_Output("汉字.txt");
                RedupRemoverLibrary.Instance.RR_Exit();
            }

        }

}
```

9.4 去重组件 C 调用示例

```
if (!RR_Init("RR_Data.txt","Data"))
    {
        printf("%s\r\n", RR_GetLastErrMsg());
        printf("去重组件初始化失败，请退出并检查！该联系 www.nlpir.org/或者新浪微博@
    ICTCLAS 张华平博士！\n");
         return-1;
    }
    vector<string> fileDir;
    string path = "./test/";
    string sContent = "";
    if(!gfn_bScanFiles(path.c_str(),fileDir))
    {
        printf("获取文件夹中文件失败.....\n");
        system("pause");
        return -1;
}
    string sFilename;
    char *pcFindAll = new char[65565];
    for (int i = 0; i < fileDir.size();i++)
    {
        gfn_bReadFile((path + fileDir.at(i)).c_str(), sContent);
        sFilename = fileDir.at(i).c_str();
        sContent += "\r\n";
```



```
        if (RR_FileProcess(sContent.c_str(), sFilename.c_str()) == 1)
        {
            memset(pcFindAll, 0, 65565);
            RR_FindRepeat(pcFindAll, true);
        }
    }
    if (pcFindAll != NULL)
    {
        delete[] pcFindAll;
        pcFindAll = NULL;
    }
    RR_Output("");
    RR_SaveData("RR_Data.txt");
```

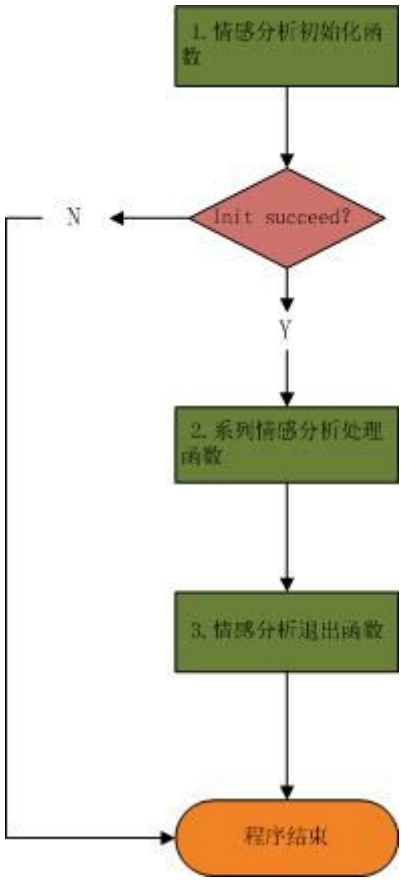
十、情感分析组件(SentimentAnalysis) 简要流程图及函数说明文档

中文情感词汇本体库是大连理工大学信息检索研究室在林鸿飞教授的指导下经过全体教研室成员的努力整理和标注的一个中文本体资源。该资源从不同角度描述一个中文词汇或者短语，包括词语词性种类、情感类别、情感强度及极性等信息。

中文情感词汇本体的情感分类体系是在国外比较有影响的 Ekman 的 6 大类情感分类体系的基础上构建的。在 Ekman 的基础上，词汇本体加入情感类别“好”对褒义情感进行了更细致的划分。最终词汇本体中的情感共分为 7 大类 21 小类。

构造该资源的宗旨是在情感计算领域，为中文文本情感分析和倾向性分析提供一个便捷可靠的辅助手段。中文情感词汇本体可以用于解决多类别情感分类的问题，同时也可以用于解决一般的倾向性分析的问题。

10.1 情感分析组件函数流程图



10.2 各个函数详细说明

10.2.1 初始化函数

① int LJST_Inits(const char *path, int encode, const char*sLicenceCode=0)

功能：初始化

备注：在进程中此函数必须在其他函数之前调用（只需执行一次）

返回：1- 成功；0 - 失败

参数： sPath - Data 文件夹的路径，为空字符串时默认从工程根目录下开始寻找

nEncoding - 编码格式，具体如下：

0：GBK；1：UTF8；2：BIG5；3：GBK（里面包含繁体字）

sLicenseCode - 授权码,写成空字符串就可以

10.2.2 系列情感分析处理函数

① bool LJST_GetParagraphSent(const char* lpszParagraph, char* szRes)

功能：获得情感分析结果

返回：true：分析成功；false：分析失败

参数：lpszParagraph - 文档内容

szRes - 分析后的结果

② bool LJST_GetFileSent(const char* lpszFilename, char* szRes)

功能：获得情感分析结果

返回：true：分析成功；false：分析失败

参数：lpszParagraph - 文档文件路径（请使用全路径的形式）

szRes - 分析后的结果

③ int LJST_ImportUserDict(const char* lpszFilename,bool bOverwrite=true)

功能：导入用户词典

备注：系统会自动处理重复词的问题

参数：sFilename - 自定义用户词典的路径。（请使用全路径的形式）

bOverwite - 是否覆盖上次添加的自定义用户词典，true：覆盖；false：不覆盖

10.2.3 情感分析退出函数

① void LJST_Exits();

功能：退出，释放资源；进程结束前须调用它释放所占用的内存资源

10.3 情感分析组件 Java 调用示例

```
 /**
 * 根据内容获得情感分析
 */
@Test
public void testGetParagraphSent(){
    //初始化
    int flag = CLibrarySentimentAnalysis.Instance.LJST_Inits("", 1, "");
    if ( flag == 0 ) {
        System.out.println("SentimentAnalysis 初始化失败");
        System.exit(0);
    } else {
        System.out.println("SentimentAnalysis 初始化成功");
    }
    //导入词典
    CLibrarySentimentAnalysis.Instance.LJST_ImportUserDict("dict/test.txt", true);
        byte[]  resultByte = new byte[10000];//分析结果
    String content = "真伪";
    //根据内容获得情感分析
    CLibrarySentimentAnalysis.Instance.LJST_GetParagraphSent(content, resultByte);
    System.out.println("根据文本内容分析结果：" + new String(resultByte));//输出分析
结果
        //退出
    CLibrarySentimentAnalysis.Instance.LJST_Exits();
    }
```

10.4 情感分析组件 C 调用示例

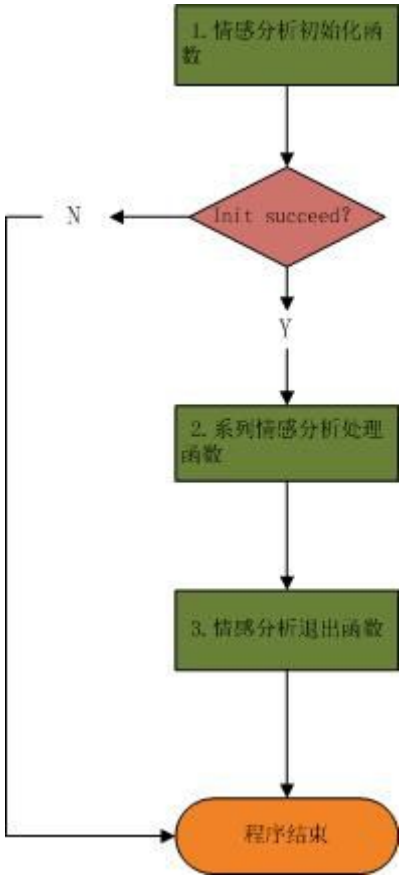
```
int values = LJST_Inits("../../",UTF8_CODE);
 if(!values)
 {
    printf("初始化失败！\n");
    return -1;
}
```

```
LJST_ImportUserDict("userdict.txt",false);//导入情感词用户词典
//LJST_ImportUserDict(const char* lpszFilename,bool bOverwrite=true);
//遍历文件夹
vector<string> fileDir;
string path = "./";
if (argc>1)
{
	path=argv[1];
}
if(!gfn_vScanFiles(path.c_str(),fileDir,"*.txt"))
{
	printf("获取文件夹中文件失败.....\n");
	return -1;
}
FILE *fp = fopen("sResult.TXT","w+");
char szRes[1024] = {0};
//printf("path : %s, filenum : %d\n",path.c_str(),fileDir.size());
for (int i = 0; i < fileDir.size()-1;i++)
{
	bool bResult = LJST_GetFileSent((path + fileDir.at(i)).c_str(),szRes);
	if (!bResult)
	{
		printf("第%d 篇文章[%s]的情感分析失败\n",i,fileDir.at(i).c_str());
		//fclose(fp);
		continue;
		//return -1;
	}
	printf("第%d 篇文章[%s]的情感类型是: %s\n",i,fileDir.at(i).c_str(),szRes);

	fprintf(fp,"第%d 篇文章[%s]的情感类型是: %s\n",i,fileDir.at(i).c_str(),szRes);
}
LJST_Exits();
fclose(fp);
```


十一、特定人物情感分析(Sentiment) 简要流程图及函数说明文档

特定人物情感分析也即倾向性分析，是指针对特定的人物（可以是多个）进行文章相关情感内容的提取。

11.1 特定人物情感分析组件函数流程图



11.2 各个函数详细说明

11.2.1 初始化函数

① bool ST_Init(const char *sInitDirPath=0, iny encode, const char* sLicenseCode="")

功能：初始化

返回：true - 成功；false - 失败

备注：在进程中此函数必须在其他函数之前调用（只需执行一次）

参数： sInitDirPath- Data 文件夹的路径，为空字符串时默认从工程根目录下开始寻找

Encode - 文本编码，具体对照如下：

0：GBK；1：UTF8；2：BIG5；3：GBK（里面包含繁体字）

sLicenseCode - 授权码,写成空字符串就可以

11.2.2 系列情感分析处理函数

① const char* ST_GetOneObjectResult(const char* sTitle, const char* sContent, const char* sObject);

功能：单条对象观点信息的分析和抽取

返回：输出结果字符串（xml 格式）

参数：sTitle - [IN] 输入文章标题

sContent - [IN] 输入文章内容

sObject - [IN] 输入分析对象名称

② const char* ST_GetMultiObjectResult(const char* sTitle, const char* sContent, const char* sObjectRuleFile);

功能：批量对象观点信息的分析和抽取

返回：输出结果字符串（xml 格式）

备注：在进程中此函数可以执行多次，支持多线程执行

参数：sTitle - [IN] 输入文章标题

sContent - [IN] 输入文章内容

sObjectRuleFile - [OUT] 输入分析对象配置文件

③ const char* ST_GetLastErrMsg();

功能：获得最近一次的错误消息

返回：最近一次错误消息

④ SENTIMENT_API int ST_SentiDictIO(const char* sPositiveFile,const char* sNegativeFile,bool bInputOutput=false);

功能：批量导入导出情感词典

返回：1:成功；0:失败

参数：sPositiveFile - [IN] 积极词典文件名；

sNegativeFile - [IN] 积极词典文件名

bInputOutput - [IN] true: 导入词典到系统中； false:导出系统词典到文本文件

11.2.3 情感分析退出函数

① void ST_Exit();

功能：退出，释放资源；进程结束前须调用它释放所占用的内存资源

11.3 特定人物情感分析组件Ｊａｖａ调用示例

```
public class LJSentimentTest
{
    /**
     * 测试导出情感词典
     */
    @Test
    public void testExportDict() {
        System.out.println("初始化开始...");
        if ( !CLibrarySentiment.Instance.ST_Init("", 1, "") ) {
            System.out.println(CLibrarySentiment.Instance.ST_GetLastErrMsg());
            System.out.println("初始化失败");
            System.exit(1);
        } else {
            System.out.println("初始化成功");
        }
        //导出积极和消极情感词典
        if ( CLibrarySentiment.Instance.ST_SentiDictIO("e:/positive.txt", "e:/negative.txt", fa
lse) == 1 ) {
            System.out.println("词典导出成功，请到对应的目录查找");
        } else {
            System.out.println("词典导出失败");
        }

        CLibrarySentiment.Instance.ST_Exit();
    }

    /**
     * 测试导入情感词典
     */
    @Test
    public void testImportDict() {
        System.out.println("初始化开始...");
        if ( !CLibrarySentiment.Instance.ST_Init("", 1, "") ) {
```

```
            System.out.println(CLibrarySentiment.Instance.ST_GetLastErrMsg());
            System.out.println("初始化失败");
            System.exit(1);
        } else {
            System.out.println("初始化成功");
        }
        //导入积极和消极情感词典
        if ( CLibrarySentiment.Instance.ST_SentiDictIO("test/pos.txt", "test/neg.txt", true) ==
= 1 ) {
            System.out.println("恭喜，词典导入成功");
        } else {
            System.out.println("词典导入失败");
        }

        CLibrarySentiment.Instance.ST_Exit();
    }

    public static void main(String[] args) {
        System.out.println("初始化开始...");
        if ( !CLibrarySentiment.Instance.ST_Init("", 1, "") ) {
            System.out.println(CLibrarySentiment.Instance.ST_GetLastErrMsg());
            System.out.println("初始化失败");
            System.exit(1);
        } else {
            System.out.println("初始化成功");
        }
        String sTitle = "郭德纲需反思：观众如水，能载舟亦能覆舟";
        String sContent = "据《法制晚报》报道，警方已经证实 1 月 9 号在首都国际机场
打人的均是德云社员工，且德云社三名员工因打人被警方处以行政拘留并罚款。目前德云
社已申请行政复议，而郭德纲与“打人”字眼再次引起人们的热议。\n " +
                "德云社为什么会频频出现打人事件，这是一个让我们很难理解的事情。在
舞台上，郭德纲把自己的身段放的那么低，与观众是那样的亲，为什么一到舞台下面就似
乎完全是换了一个人呢?\n" +
                "郭德纲不在体制内，按他的话说自己就是一个“非著名相声演员”，而那些
在体质内的则统一被他戏称为“主流的”，而且在一切场合尽自己的最大的可能来讽刺和挖
苦这些所谓的“主流”相声演员。郭德纲成名，靠的不是哪个政府部门，靠的是自己坚持不
懈的努力，靠的是观众们的力捧，靠的是电视台网络的大力宣传。所以，他在唱经典段子
《大实话》的时候会一直唱“要说亲，观众们亲，观众演员心连着心!”";

        System.out.println(CLibrarySentiment.Instance.ST_GetOneObjectResult(sTitle, sConte
nt, "郭德纲"));
        System.out.println("==========================================
=======================================");
        System.out.println(CLibrarySentiment.Instance.ST_GetMultiObjectResult(sTitle, sCon
```



```
tent, "stConduct.xml"));

        CLibrarySentiment.Instance.ST_Exit();

        System.exit(0);
    }
}
```

11.4 特定人物情感分析 C 调用示例

```
int main(int argc, char* argv[])
{
    if(!ST_Init("D:\\NLPIR\\Data\\Sentiment\\")){
        printf("%s", ST_GetLastErrMsg());
        getchar();
        return 1;
    }
    string sTitle = "郭德纲需反思：观众如水，能载舟亦能覆舟";
    string sContent = "据《法制晚报》报道，警方已经证实 1 月 9 号在首都国际机场打人的均是德云社员工，且德云社三名员工因打人被警方处以行政拘留并罚款。目前德云社已申请行政复议，而郭德纲与“打人”字眼再次引起人们的热议。\n\
                德云社为什么会频频出现打人事件，这是一个让我们很难理解的事情。在舞台上，郭德纲把自己的身段放的那么低，与观众是那样的亲，为什么一到舞台下面就似乎完全是换了一个人呢?\n\
                郭德纲不在体制内，按他的话说自己就是一个“非著名相声演员”，而那些在体质内的则统一被他戏称为“主流的”，而且在一切场合尽自己的最大的可能来讽刺和挖苦这些所谓的“主流”相声演员。郭德纲成名，靠的不是哪个政府部门，靠的是自己坚持不懈的努力，靠的是观众们的力捧，靠的是电视台网络的大力宣传。所以，他在唱经典段子《大实话》的时候会一直唱“要说亲，观众们亲，观众演员心连着心!”";
    printf("【标题】：%s\n【正文】：%s\n", sTitle.c_str(), sContent.c_str());
    printf("\n**********\n");
    const char* pResult = ST_GetMultiObjectResult(sTitle.c_str(), sContent.c_str(), "stConduct.xml");
    printf("\n 批量分析 stConduct.xml：\n\n%s\n", pResult);
    printf("\n**********\n");
    pResult = ST_GetOneObjectResult(sTitle.c_str(), sContent.c_str(), "郭德纲");
    printf("\n 单独分析“郭德纲”：\n\n%s\n", pResult);
    ST_Exit();
    getchar();
    return 0;
}
```


十二、精准搜索组件(JZSearch) 简要流程图及函数说明文档

精准搜索组件内核经过精心设计，具有高扩展性和高通用性，可支持文本、数字、日期、字符串等各种数据类型的高效索引，支持丰富的查询语言和查询类型，支持少数民族语言的搜索。同时，全文搜索中间件可以无缝地与现有数据库系统融合，实现全文搜索与相关的数据库管理应用系统。其主要特色在于：

* 可以按照任意指定字段的排序，支持指定字段的搜索，也可以搜索多个字段，以及复杂表达式的综合搜索；

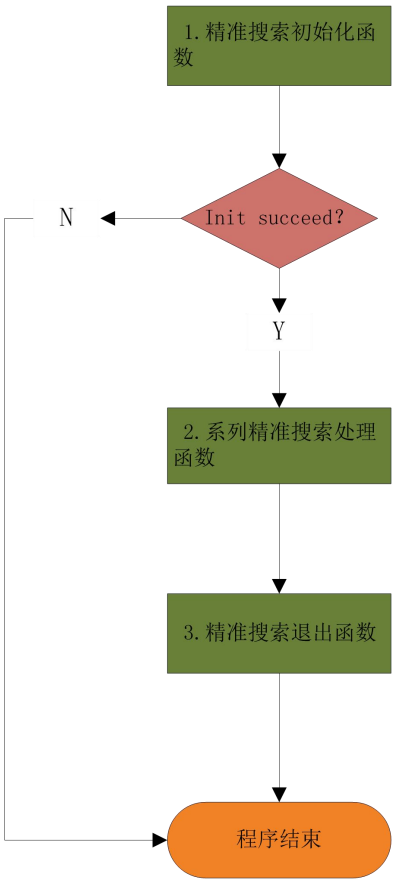
* 支持精确匹配以及模糊匹配，默认为精确匹配，忽略字母大小写进行模糊匹配；

* 实现的是多线程搜索服务；

* 每秒可索引 3000 条记录（主要瓶颈为数据库或文件记录的读取效率）；搜索速度在毫秒级别。

* 兼容当前所有厂商的数据库系统，其中 SQL Server, Oracle, MySQL，DB2 等。

12.1 精准搜索组件函数流程图



12.2 各个函数详细说明

12.2.1 初始化函数

① bool JZSearch_Init(const char *sDictPath=0,const char *sFieldInfoFile=0,int encoding=INDEX_ENCODING_GBK_FANTI,bool bICTCLAS=false,int lang_type=LANGUAGE_TYPE_CHINESE,const char*sLicenceCode=NULL);

功能：索引系统初始化，成功返回 true，失败返回 false；

sDictPath：字典文件所在路径，默认在当前文件夹下；

encoding：编码格式，有 utf-8，big5 等；

sLicenceCode：授权码，如果是一周授权默认为空；

12.2.2 组件系列处理函数

① bool JZSearch_Reload(SEARCHER_HANDLE handle=0);

功能：搜索器重新加载函数，成功返回 true，失败返回 false；

Handle：搜索器句柄，默认为 0 表示为第一个搜索器；

② const char * JZSearch_ReloadBlackList(void);

功能：搜索器重新加载黑名单函数，成功返回黑名单的索引指针，失败返回 NULL；

③ const char * JZSearch_ReloadQueryExpand(void);

功能：重新加载扩展词更新函数，成功返回索引指针，失败返回 NULL；

④ int JZSearch_ListFieldAllValue(const char *sFieldLine,const char *sExportFile,bool bDuplicateErase=false,SEARCHER_HANDLE handle=1);

功能：列出指定字段的内容；

sFieldLine：索引指针；

sExportFile：指定字段输出指针；

Handle：索引器编号；

⑤ bool JZSearch_Export(const char *sExportFile,const char *sWordList,char arg,SEARCHER_HANDLE handle=0);

功能：导出内部的数据，以便核查；

sExportFile：导出文件句柄；

sExportFile：词组列表；

Handle：索引器编号；

⑥ bool JZSearch_Merge(SEARCHER_HANDLE handle=0);

功能：系统自动优化，对索引信息进行归并处理，成功返回 true，失败返回 false；

⑦ bool JZSearch_Backup(const char *sFilename=0,SEARCHER_HANDLE handle=0);

功能：系统索引数据自动备份，一般不建议用户自行使用，系统会自动备份；

sFilename：索引文件名；

⑧ bool JZSearch_Restore(SEARCHER_HANDLE handle=0);

功能：系统索引数据自动回复，一般不建议用户自行使用，系统会在无法加载当前数据时自动恢复；

12.2.3 退出函数

① bool JZSearch_Exit();

功能：退出，释放资源；进程结束前须调用它释放所占用的内存资源

12.3 精准搜索组件 Java 调用示例

```
public class LJSearchServer {
    private String ip;
    private int port;
    public LJSearchServer(String ip, int port) {
        this.ip = ip;
        this.port = port;
    }
    public LJSearchServer()
{
    }

    public synchronized LJSearchResult search(String query, int start, int size) throws
IOException  {
        if (query == null || query.trim().length() == 0) {
            return null ;
        }
        Socket socket = null;
        DataOutputStream dos = null ;
        DataInputStream dis = null ;
        try {
```



```
//                  byte[] bytes = query.getBytes("GBK");
                    byte[] bytes = query.getBytes("UTF-8");
                    socket = new
Socket(ParamProperty.getValue("search_ip"),Integer.parseInt(ParamProperty.getValue("search_po
rt").trim()));
//                  socket = new Socket(ip,port);
                    socket.setSoTimeout(60000) ;
                    dos = new DataOutputStream(
                            socket.getOutputStream());
                    dis = new DataInputStream(socket.getInputStream());
                    //c 内存中有三个 int 要返回所以+12，
                    dos.writeInt(bytes.length+12);
//                  dos.writeInt(5);
                    dos.writeInt(start);
                    dos.writeInt(size);
                    dos.writeInt(bytes.length);
                    dos.write(bytes);
                    if(size>0){
                        LJSearchResult ljSearchResult = new LJSearchResult(dis) ;
                        //读取 socket 流内容
                        ljSearchResult.readSocket();
                        return ljSearchResult ;
                    }else{
                        return null ;
                    }
                }catch(Exception e ){
                    e.printStackTrace() ;
                } finally {
                    dos.close();
                    dis.close();
                    socket.close();
                }
                return null;
        }
        public static void main(String[] args) throws IOException {
String str="北京/14#中国/13#王菲/10#信息/9#信息安全/6#电子/6#认证/5#培训/5#国家/5#沙
钢船务/5#";
                new LJSearchServer("192.168.1.202",10000).search("[FIELD] * [AND]  北京
[CLUSTER]    Keywords", 76, 90) ;
        }
}
```


12.4 精准搜索组件 C 调用示例

```
Bool bRtn=JZSearch_Init("D:\\NLPIR\\feedback\\标院索引\\IndexFile\\JZSearch","D:\\NLPIR\\release\\JZSearchInstall\\dict","D:\\NLPIR\\feedback\\标院索引\\IndexFile\\field.dat");
    if (!bRtn)
    {
        printf("JZSearch Init failed!\n");
        return 0;
    }
    CJZSearcher *pSearcher=new CJZSearcher(SORT_TYPE_DOCID);//按照相关度排序
    char sLine[1000];
    printf("Please input search cmd\n");
    fgets(sLine,1000,stdin);     /*pSearcher->DocDelete(3908);pSearcher->DocDelete(2110); */
    clock_t start, finish;
    double duration;
    int nCount;
    std::vector<int> vecDocID;
    int nResultSize=0,i;
    const char*pItems;
    while (sLine[0]!='q'&&sLine[0]!='Q')
    {
        start = clock();
        nResultSize=pSearcher->Search(sLine,vecDocID);
        for (i=0;i<nResultSize;i++)
        {
            printf("Result docid=%d:\n",vecDocID[i]);
            pItems=pSearcher->GetFiledValue(vecDocID[i],"pdflink");
            //get the field "pdflink" value
            printf("pdflink=%s\n",pItems);
```

```
            pItems=pSearcher->GetFiledValue(vecDocID[i],"pdflink3");//get the field "pdfli
nk" value //if the field name is invalid return NULL
            printf("pdflink3=%s\n",pItems);
pItems=pSearcher->GetFiledValue(vecDocID[i],"a100");//get the field "pdflink" value
            printf("a100=%s\n",pItems);
pItems=pSearcher->GetFiledValue(vecDocID[i],"stdclassid");//get the field "pdflink" value
            printf("stdclassid=%s\n",pItems);
      pItems=pSearcher->GetFiledValue(vecDocID[i],"appendumtitle");//get the field "pdflink"
value
            printf("appendumtitle=%s\n",pItems);
        }
        //pSearcher->Search(sLine,0,200,"ResultTemp.xml");//搜索 sQuerySearch
        //pSearcher->Search(sLine,&nCount);//搜索 sQuerySearch
        //pSearcher->GetResultBuf()
        finish = clock();
        duration = (double)(finish - start) / CLOCKS_PER_SEC;
        printf("Time=%f\n",duration);
        printf("Please input search cmd\n");
        fgets(sLine,1000,stdin);
    }
    delete pSearcher;
    JZSearch_Exit();
```


十三、作者简介

张华平 博士 副教授 研究生导师
北理工大数据搜索挖掘实验室 主任
地址：北京海淀区中关村南大街 5 号 100081
手机：13681251543
电话：+86-10-68918642
Email:kevinzhang@bit.edu.cn
MSN: pipy_zhang@msn.com;
网站: http://www.nlpir.org (自然语言处理与信息检索共享平台)
http://www.bigdataBBS.com (大数据论坛)
博客:http://hi.baidu.com/drkevinzhang/
微博:http://www.weibo.com/drkevinzhang/
Dr. Kevin Zhang (张华平，Zhang Hua-Ping)
Associate Professor, Graduate Supervisor
Director, Big Data Search and Mining Lab.
Beijing Institute of Technology
Add: No.5, South St.,Zhongguancun,Haidian District,Beijing,P.R.C PC:100081
Tel: +86-10-68918642
Email:kevinzhang@bit.edu.cn
MSN: pipy_zhang@msn.com;
Website: http://www.nlpir.org (Natural Language Processing and Information Retrieval Sharing Platform)
http://www.bigdataBBS.com (Big Data Forum)
Blog:http://hi.baidu.com/drkevinzhang/
Twitter: http://www.weibo.com/drkevinzhang/

