



NLPIR 大数据搜索与挖掘共享开发平台

NLPIR Big Data Search and Mining Development Platform

白皮书

WHITEPAPER



April 16, 2013

For the latest information about NLPIR, please visit [Http://ICTCLAS.nlpir.org/](http://ICTCLAS.nlpir.org/)

Document Information

Document ID	NLPIR-ICTCLAS-2013-WHITEP APER	Version	V3.0
Security level	Public 公开	Status	Creation and first draft for comment
Author	张华平	Date	Jan 5, 2012
Publisher	/	Approved by	

Version History

Note: The first version is "v0.1". Each subsequent version will add 0.1 to the exiting version. The version number should be updated only when there are significant changes, for example, changes made to reflect reviews. The first figure in the version 1.x denotes current review status by. 1. x denotes review process has passed round 1 etc .Anyone who create, review or modify the document should describe his action.

Ver sio n	Author/Re viewer	Date	Description
V1. 0	Kevin Zhang	2011-8-21	first complete draft for comment. ICTCLAS2010
V2. 0	Kevin Zhang	2012-8-21	complete draft for comment.ICTCLAS2012
V3. 0	Kevin Zhang	2012-12-19	complete draft for comment.ICTCLAS2013



目 录

一、NLPIR 大数据搜索与挖掘共享开发平台简介.....	4
1 汉语词法分析中间件（分词、词性标注、人名地名机构名识别）.....	4
2 新语自动发现中间件.....	6
3 文本内容去重中间件.....	9
4 文本分类过滤中间件.....	11
5 文本聚类中间件.....	13
6 文档关键词提取中间件.....	17
7 文本摘要中间件.....	19
8 正负面分析中间件.....	20
9 网页正文提取中间件.....	23
10 全文搜索中间件.....	24
二、作者简介.....	27

一、NLPIR 大数据搜索与挖掘共享开发平台简介

NLPIR 大数据搜索与挖掘共享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：

1 汉语词法分析中间件（分词、词性标注、人名地名机构名识别）

汉语词法分析中间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。NLPIR 大数据搜索与挖掘共享开发平台综合了各家所长，采用条件随机场（Conditional Random Field,简称 CRF）模型，分词准确率接近 99%，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”。NLPIR 大数据搜索与挖掘共享开发平台采用条件随机场（Conditional Random Field,简称 CRF）模型，一级词性标注准确率接近99%，具备准确率高、速度快、可适应性强等优势。

人名地名机构名识别能够自动挖掘出隐含在汉语中的人名、地名、机构名，所提炼出的词语不需要在词典库中事先存在，是对语言规律的深入理解和预测。采用条件随机场（Conditional Random Field,简称 CRF）模型，识别准确率达到

97%，速度达到 10M/s，可在此基础上搭建各种多样化的统计和应用。

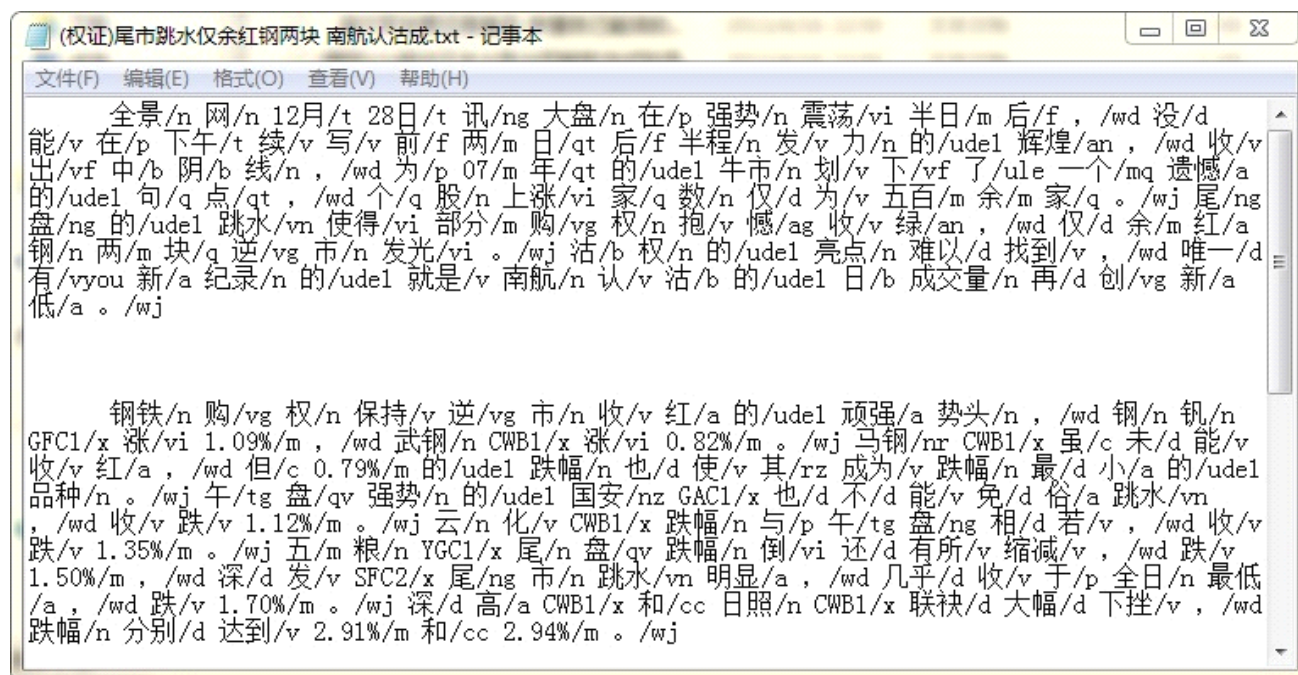


图 1 汉语词法分析效果展示

主要接口：

- * Description: The function must be invoked before any operation listed as following
- * Parameters : const char * sInitDirPath=NULL
- * sInitDirPath: Initial Directory Path, where file Configure.xml and Data directory stored.
- * the default value is NULL, it indicates the initial directory is current working directory path
- * Returns : success or fail

*****/

CRF_API bool CRF_Init(const char * sInitDirPath=0);

- * Description: Exist and free related buffer, the function must be invoked while you needn't any lexical anlysis
- * Parameters : None
- * Returns : success or fail

*****/

```
CRF_API bool CRF_Exit();
```

```
/******
```

```
* Func Name : ImportUserDict
```

```
* Description: Import User-defined dictionary
```

```
* Parameters : Text filename for user dictionary
```

```
*****/
```

```
CRF_API unsigned intCRF_ImportUserDict(const char *sFilename);
```

```
/******
```

```
* Func Name : ParagraphProcessing
```

```
* Description: Process a paragraph
```

```
* Parameters : sParagraph: The source paragraph
```

```
* sResult: The result
```

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

```
* i.e. 张春阳于 1982 年 3 月 9 日出生于辽宁省辽阳市。
```

```
* Result: 张/nr 春阳/nr 于/p 1982 年/t 3 月/t 9 日/t 出生于/v 辽宁省/ns 辽阳市/ns 。/w
```

```
* Returns : the result buffer pointer
```

```
*****/
```

```
CRF_API const char * CRF_ParagraphProcess(const char *sParagraph,int bPOStagged=1);
```

2 新语自动发现中间件

新词自动发现技术能够识别出词典中没有出现过的词汇、短语、命名实体、流行用语，是语言文献分析方面的一把利器。新词发现脱胎于语言自动分词技术，又是对分词技术的有效提升和补充。

NLPIR 大数据搜索与挖掘共享开发平台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，能够有效地挖掘出新出现的特征词汇，所输出的词汇可以配以权重。

新词发现中间件的主要特色在于：

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 60 万篇文档；

2、处理精准：Top N 的分析结果往往能反映出当时的时事流行语和热点实体，适合于舆情热点计算；与国际上著名厂商的技术相比，各项指标远远领先，或许是 nlpir 更懂中文吧；

3、精准排序：新词汇按照影响权重排序，可以输出权重值；

5、开放式接口：新词发现组件作为 NLPIR 大数据搜索与挖掘共享开发平台的一部分，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。

新词发现组件可以应用于文本挖掘、知识管理、词典编辑、舆情监测等多种应用中。

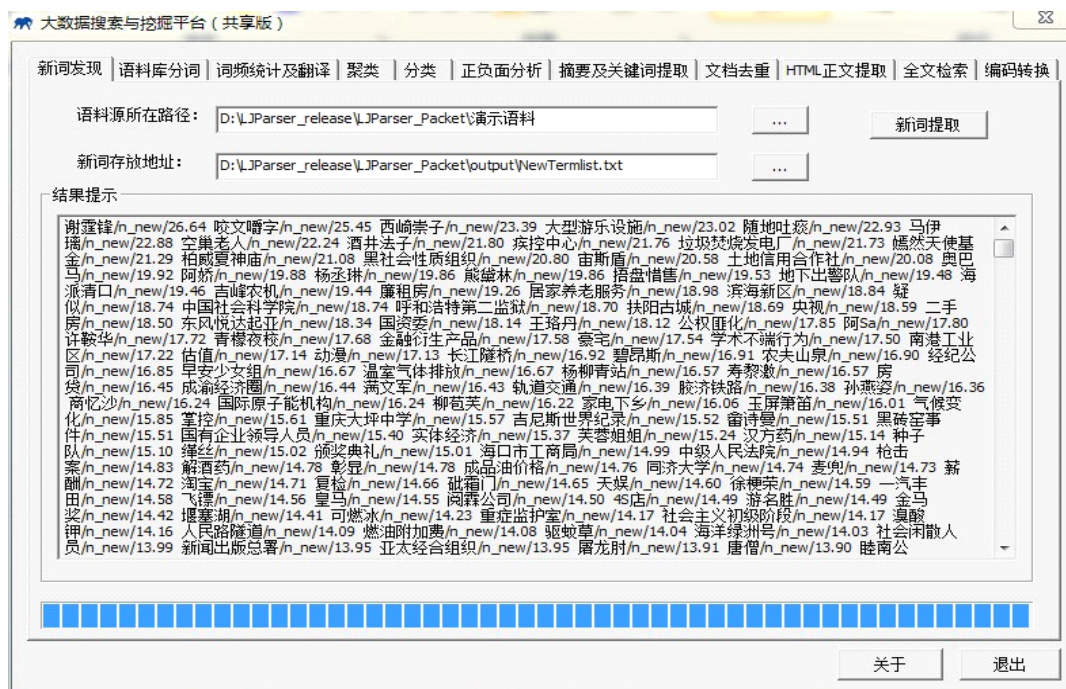


图 2 语料新词自动发现结果展示

主要接口：

```
/*-----  
  
* 功能：初始化  
  
* 参数：sLicenseCode - 授权码  
  
* 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）  
-----*/  
  
FEATUREDETECT_API bool WDT_Init(const char *sLicenseCode=0);  
  
/*-----  
  
* 功能：追加内存内容  
  
* 参数：sText          - [IN] 正文内容（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特征词抽取之前执行多次  
-----*/  
  
FEATUREDETECT_API bool WDT_AddContent(const char *sText);  
  
/*-----  
  
* 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）  
  
* 返回：true - 成功；false - 失败  
  
* 备注：在进程中此函数可以在特征词抽取之前执行多次  
-----*/  
  
FEATUREDETECT_API bool WDT_AddFile(const char *sFileName);  
  
/*-----  
  
* 功能：取得最新结果  
  
* 参数：nMaxCount       - [IN] 最多抽取多少个关键词
```


* pnRealCount - [OUT] 实际抽取出的关键词个数

* bWeightFlag - [IN] 是否输出权重

* 返回：特征词字符串（以\t分隔，词和权重间用空格分隔）

-----*/

```
FEATUREDETECT_API const char* WDT_GetLatestResult(int nMaxCount, int &nRealCount, bool  
bWeightFlag=false);
```

//清空历史数据

```
FEATUREDETECT_API void WDT_CleanData();
```

//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

```
FEATUREDETECT_API void WDT_Exit();
```

//获得错误消息

```
FEATUREDETECT_API const char* WDT_GetLastErrMsg();
```

3 文本内容去重中间件

文本内容去重中间件能够对文本进行查重处理，同时能找出所有的重复文件。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。NLPIR 大数据搜索与挖掘共享开发平台采用高效的文章指纹算法，能够在极短的时间内与历史指纹库进行对比，从而发现重复记录。

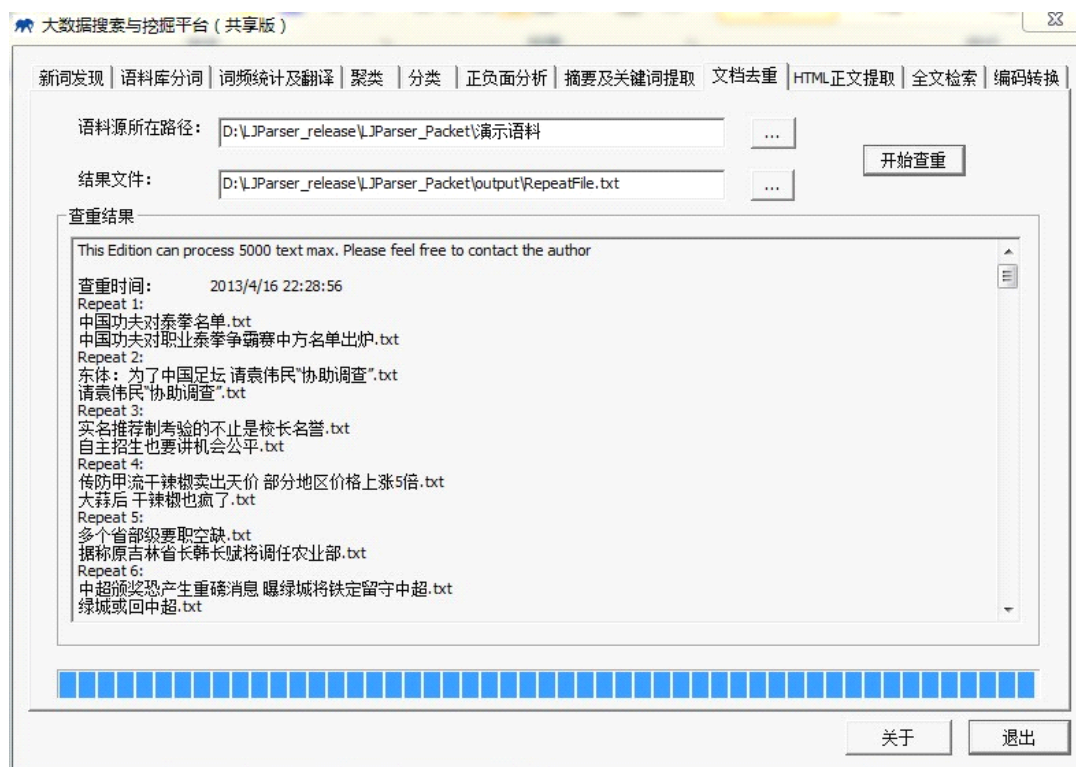


图 3 内容去重业务功能展示效果

主要接口:

//初始化,并装载已有的查重库数据

```
FileFilter_API bool RR_Init(const char *DataFilePath = "Data.txt", const char *sLicenseCode=0);
```

//退出

```
FileFilter_API void RR_Exit();
```

//查看当前内容是否为重复文本

```
FileFilter_API int RR_FileProcess(const char *FileText, const char *FileName);
```

//查看当前内容具体与哪些文本存在重复关系

```
FileFilter_API char * RR_FindRepeat(bool isAll = false);
```

//保存重复文本信息

```
FileFilter_API bool RR_Output(const char *OutputPath = "RepeatFile.txt");
```

```
//保存查重库数据
```

```
FileFilter_API bool RR_SaveData(const char *DataFilePath = "Data.txt");
```

```
//获得错误提示消息
```

```
FileFilter_API const char* RR_GetLastErrMsg();
```

4 文本分类过滤中间件

文本分类中间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。

文本过滤功能能够从大量文本中快速识别和过滤出符合特殊要求的信息，可应用于品牌报道监测、垃圾信息屏蔽、敏感信息审查等领域。

NLPIR 大数据搜索与挖掘共享开发平台采用基于内容的文本自动分类过滤和基于规则的文本分类过滤两种方式，并支持两种方式的混合分类。能够进行多级分类，分类速度每秒 100 篇以上，平均准确率 90%以上，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。

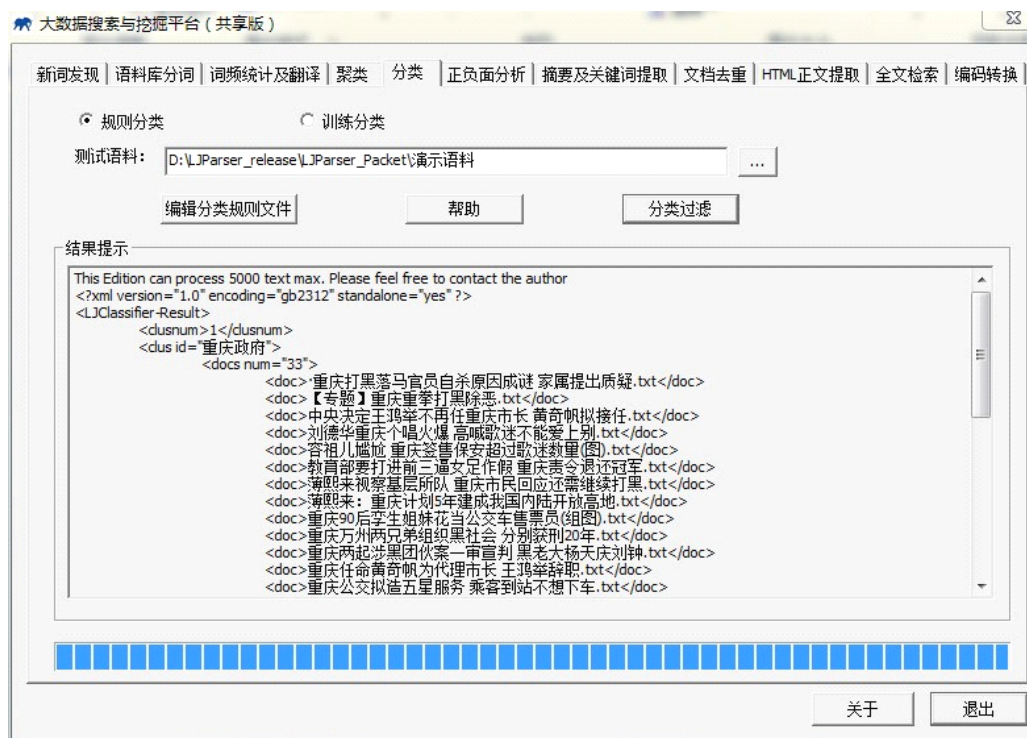


图 4 分类相关业务功能展示效果

主要接口：

// 功能： 文件方式初始化

// 返回值： 成功/失败

```
CLASSIFIER_API bool classifier_init(const char *conf="rulelist.xml", const char *sLicenseCode=0);
```

// 功能：对输入的文章结构进行分类

// 参数：d:文章结构指针

// 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 敏感诉讼 0.82”

// 返回值：主题类别串 各类之间用\t 隔开，类名按照置信度从高到低排序

```
CLASSIFIER_API const char* classifier_exec(stDoc* d, int iType=0);
```

// 功能：对于当前文档，输入类名，取得结果明细

// 参数：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 */

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

// 功能：退出，释放资源

CLASSIFIER_API void classifier_exit();

5 文本聚类中间件

文本聚类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。

NLPIR 大数据搜索与挖掘共享开发平台基于文章集合核心语义理解技术，不仅聚类速度快，而且准确率高，并能自动得到类别间的演化趋势。

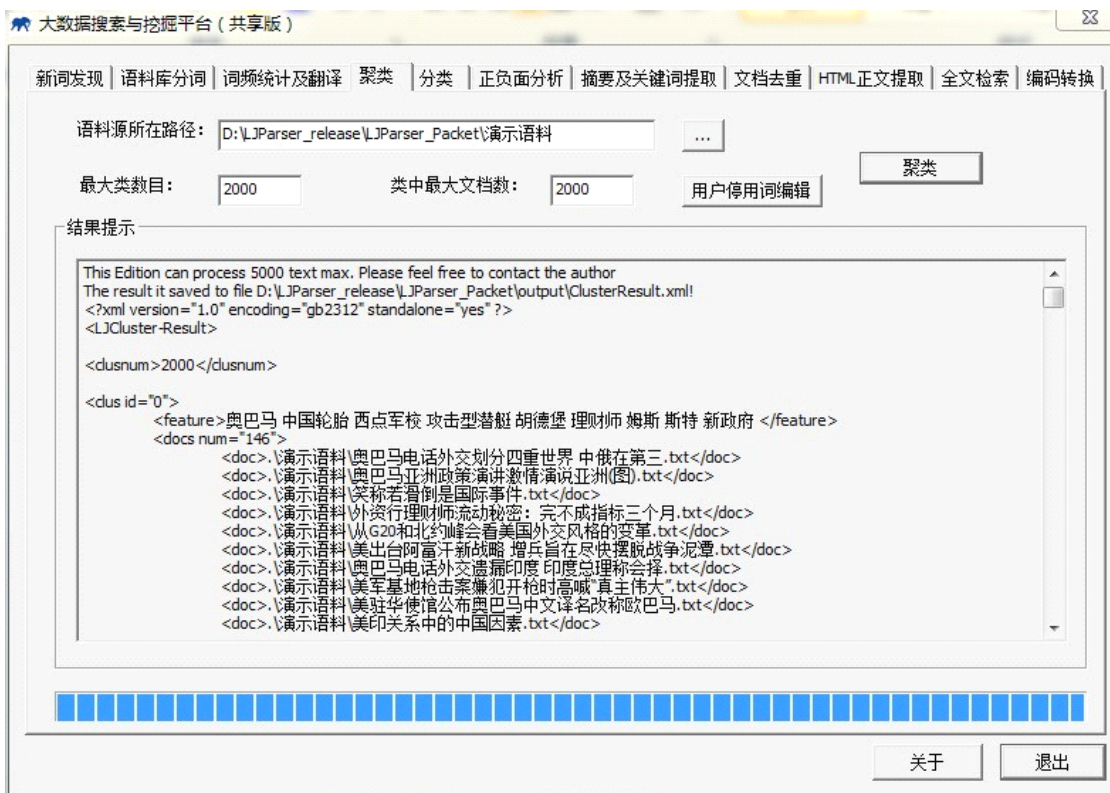


图 5 聚类特征和文章集合展现图

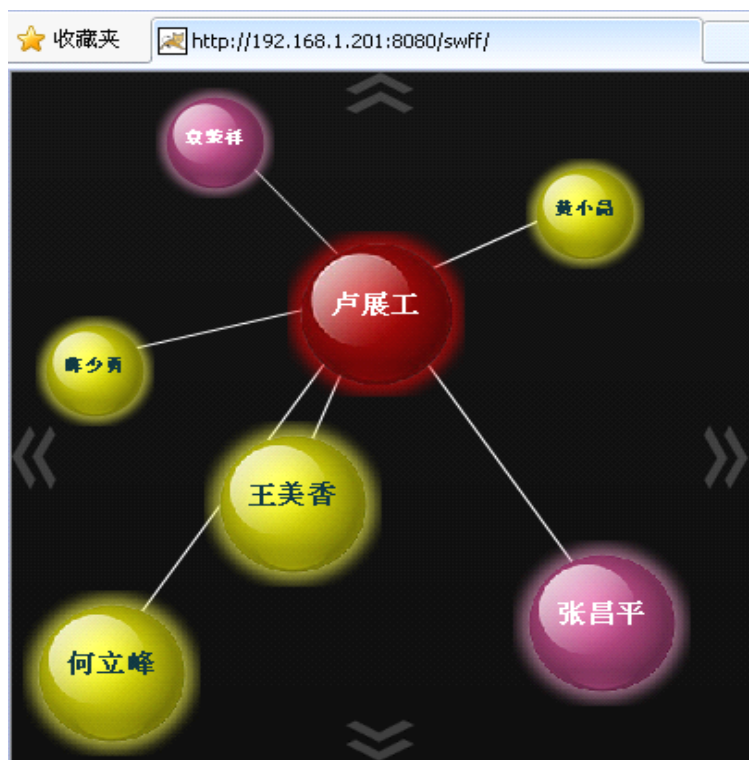


图 6 热点人物关联展现图

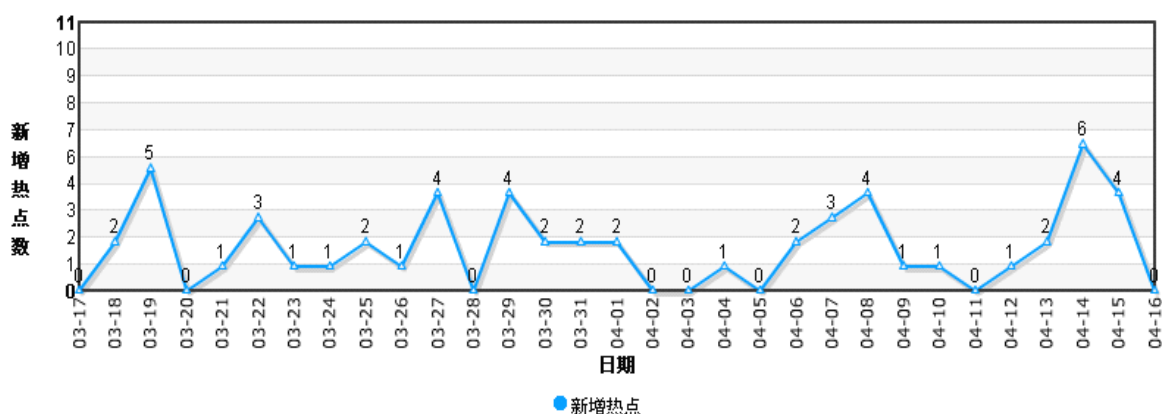


图7 新增热点话题趋势统计图

主要接口：

```
/*-----
```

* 功能：初始化

* 参数：sLicenseCode - 授权码

* 返回：true - 成功；false - 失败

* 备注：在进程中此函数必须要在其他函数之前调用（只需执行一次）

```
-----*/
```

```
CLUSTER_API bool CLUS_Init(const char *sLicenseCode=0);
```

```
/*-----
```

* 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要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过序

```
-----*/
```

```
CLUSTER_API bool CLUS_SetParameter(int nMaxClus, int nMaxDoc);
```

```
/*-----
```

* 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

-----*/

CLUSTER_API bool CLUS_AddContent(const char *sText, 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

-----*/

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

/*-----

* 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

-----*/

CLUSTER_API bool CLUS_GetLatestResult(const char* sXmlFileName);

//清空历史数据

CLUSTER_API void CLUS_CleanData();

//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

```
CLUSTER_API void CLUS_Exit();
```

```
//获得错误消息
```

```
CLUSTER_API const char* CLUS_GetLastErrMsg();
```

6 文档关键词提取中间件

文章关键词提取中间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 NLPIR 大数据搜索与挖掘共享开发平台的一部分，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。



图 8 关键词提取业务功能展示效果

主要接口：

```
/*-----*/
```

* 功能：初始化

* 参数：sLicenseCode - 授权码

* 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）

```
-----*/
```

```
KWEXTRACT_API bool KDT_Init(const char *sLicenseCode=0);
```

```
/*-----*/
```

* 功能：分析文本内容

* 参数：sText - [IN] 文本内容（以'\0'结束的字符串）

* nMaxCount - [IN] 最多抽取多少个关键词

* bWeightFlag - [IN] 是否输出权重

* 返回：特征词字符串（以\t分隔，词和权重间用空格分隔）；出错返回空串

* 备注：在进程中此函数可以在特征词抽取之前执行多次

-----*/

```
KWEXTRACT_API const char* KDT_ParseContent(const char *sText, int nMaxCount, bool  
bWeightFlag=false);
```

//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

```
KWEXTRACT_API void KDT_Exit();
```

//获得错误消息

```
KWEXTRACT_API const char* KDT_GetLastErrMsg();
```

7 文本摘要中间件

自动文本摘要中间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。

自动摘要中间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。

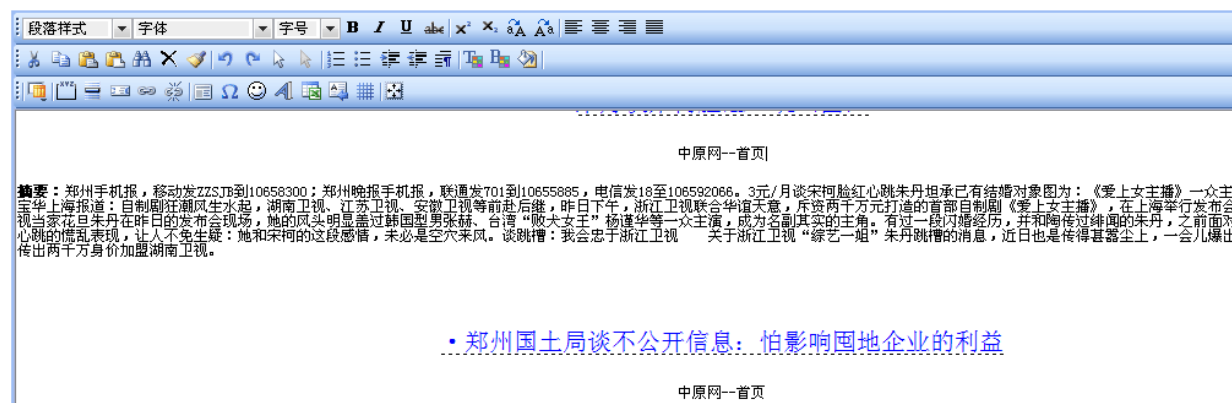


图 9 自动摘要业务功能展示效果

主要接口：

```
/*-----  
* 功能：初始化  
* 参数：sLicenseCode - 授权码  
* 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）  
-----*/  
DOCSUMMARY_API bool DS_Init(const char *sLicenseCode=0);  
  
/*-----  
* 功能：生成单文档摘要  
* 参数：sText      -[IN] 文档内容  
*          fSumRate-[IN] 文档摘要占原文百分比（为 0.00 则不限制）  
*          iSumLen  -[IN] 用户限定的摘要长度（为 0 则不限制）  
* 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  
-----*/  
DOCSUMMARY_API const char* DS_SingleDoc(const char *sText, float fSumRate=0.00, int iSumLen=250);  
  
//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  
DOCSUMMARY_API void DS_Exit();  
  
//获得错误消息  
DOCSUMMARY_API const char* DS_GetLastErrMsg();
```

8 正负面分析中间件

正负面分析中间件能够实现文本内容的观点分析，从长篇文章中自动提取指

定分析对象的正负面信息，构成观点摘要，方便用户快速对分析对象清晰了解。

正负面分析中间件不仅可以针对单独对象进行深入分析，还能够将批量对象进行分析；在使用系统默认词库基础上，用户可以自由设定自己的个性化词库。可用于商品声誉的网上追踪，某人或某事的正负面报道，公司声望的网上追踪，重大事件的民意自动调查，各类事务的观点曲线等。

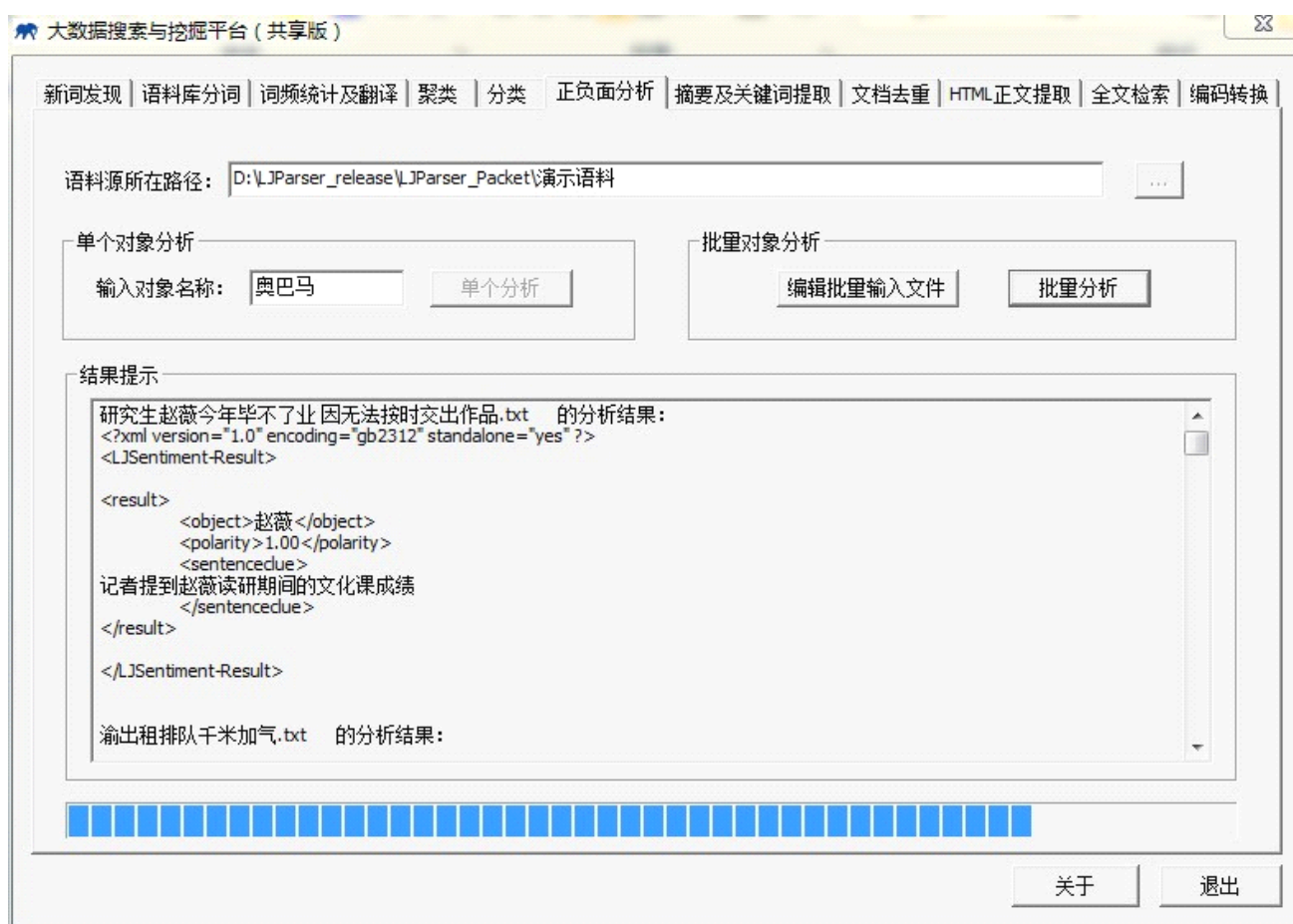


图 10 正负面分析业务功能展示效果

主要接口：

/*-----

* 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）

-----*/

SENTIMENT_API bool ST_Init(const char* sLicenseCode=NULL);

/*-----

* 功能：观点信息的分析和抽取（单条）

* 参数：sTitle - [IN] 输入文章标题

* sContent - [IN] 输入文章内容

* sObject - [OUT] 输出结果文件路径

* 返回：输出结果字符串

* 备注：在进程中此函数可以执行多次，支持多线程执行

-----*/

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

/*-----

* 功能：观点信息的分析和抽取（多条）

* 参数：sTitle - [IN] 输入文章标题

* sContent - [IN] 输入文章内容

* sObject - [OUT] 输出结果文件路径

* 返回：输出结果字符串

* 备注：在进程中此函数可以执行多次，支持多线程执行

-----*/

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

//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

```
SENTIMENT_API void ST_Exit();
```

```
//获得最近一次的错误消息
```

```
SENTIMENT_API const char* ST_GetLastErrMsg();
```

9 网页正文提取中间件

网页正文提取中间件能够实现 HTML 格式的精确分析，自动分辨出网页是属于索引页面还是内容页面。对于内容页面，能够高效剔除 HTML 标签和导航、广告等干扰性文字，返回实际有价值的正文内容。特别适用于大规模互联网信息的高效预处理和分析。

NLPIR 大数据搜索与挖掘共享开发平台基于统计分布规律模型判定网页的特征，所适用的网页不受类型和领域的限制，不需要配置抽取规则，能够全自动地对海量互联网信息信息进行高效处理。

业务功能展示效果详见图 8。

主要接口：

```
//初始化
```

```
HPARSER_API bool HPR_Init(const char *sLicenseCode);
```

```
//退出，释放资源
```

```
HPARSER_API void HPR_Exit();
```

```
//对 HTML 进行解析（只有执行此函数后才能 HPR_GetContent）
```

```
HPARSER_API bool HPR_ParseFile(const char *sHtmlFilename);
```

```
HPARSER_API bool HPR_ParseBuffer(const char *sHtmlBuffer, int nLen);
```

//提取正文，返回 NULL 时表示失败（调用 TE_GetLastErrMsg 可获得错误提示）

HPARSER_API const char* HPR_GetContent();

//获得错误提示消息

HPARSER_API const char* HPR_GetLastErrMsg();

10 全文搜索中间件

全文搜索中间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。



图 11 全文搜索中间件业务展示效果

主要接口:

////////////////////////////////////

// 以下部分为索引 API

////////////////////////////////////

//索引系统初始化

//sDictFilename: 词典文件名; 为空时, 采用 n-gram 索引方法

LJSEARCHAPI_API bool LJIndexer_Init(const char *sDictFilename=0);

//系统退出

LJSEARCHAPI_API bool LJIndexer_Exit();

//索引合并, sIndexFile1 的 doc_id 编号均小于 sIndexFile2 的 doc_id

LJSEARCHAPI_API bool LJIndexer_Merge(const char *sIndexFile1,const char *sIndexFile2,const char *sIndexMerged);

```
//建立索引的类

class LJSEARCHAPI_API CLJIndexer {

public:

//内存大小控制

CLJIndexer(int nMaxMemSize=512000000);

bool MemIndexing(const char *pText,int doc_id,unsigned char nFiledID=0xff,int nMemSize=0);//索引一段内存,
doc_id 由应用程序维护

bool FileIndexing(const char *sTextFilename,int doc_id,unsigned char nFiledID=0xff);//索引一个文本文件,
doc_id 由应用程序维护

bool IdIndexing(int term_id,int doc_id,unsigned char nFiledID=0xff);//词 ID 索引,doc_id 由应用程序维护

bool Save(const char *sIndexFile);//索引保存的名称

}

////////////////////////////////////

// 以下部分为搜索 API

////////////////////////////////////

//检索系统初始化

//sDictFilename: 词典文件名; 为空时, 采用 n-gram 索引方法

//sIndexFile: 索引文件

LJSEARCHAPI_API bool LJSearch_Init(const char *sIndexFile,const char *sDictFilename=0);

//系统退出

LJSEARCHAPI_API bool LJSearch_Exit();

//搜索结果结构, 用于检索计算使用

typedef struct tRESULT_RECORD {

    int doc_id;

    int offset;//在域字段内的偏移量
```



```
double score;//排序用的打分机制

}RESULT_RECORD;

typedef RESULT_RECORD * RESULT_RECORD_VECTOR;

class LJSEARCHAPI_API CLJSearcher{

public:

    CLJSearcher(int sort_type=SORT_TYPE_DOCID);

    //索引保存的名称，检索之前，必须 Load 索引文件

    bool Load(const char *sIndexFile);

    //搜索关键词，返回搜索结果数目；

    //结果存储在 m_pResult 指针内，无需修改，只读即可

    const RESULT_RECORD_VECTOR Search(const char *sKeyword,int *p_nResultCountRet,int nSize=0);

};
```

二、作者简介



张华平 博士 副教授 硕导
北京理工大学计算机学院 院长助理
北京理工大学网络搜索挖掘与安全实验室 主任
地址：北京海淀区中关村南大街 5 号 100081
电话：+86-10-68918642
Email: kevinzhang@bit.edu.cn
MSN: pipy_zhang@msn.com;



网站: <http://ictclas.nlpir.org> (自然语言处理与信息检索共享平台)

博客: <http://hi.baidu.com/drkevinzhang/>

微博: @ICTCLAS 张华平博士

Dr. Kevin Zhang (张华平, Zhang Hua-Ping)

Associate Professor, Graduate Supervisor

Dean Assistant, School of Computer

Director, Web Search, Mining and Security 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: pihy_zhang@msn.com;

Website: <http://ictclas.nlpir.org> (Natural Language Processing and Information Retrieval Sharing Platform)

Blog: <http://hi.baidu.com/drkevinzhang/>

Twitter: @ICTCLAS 张华平博士