

# IKAnalyzer 中文分词器

## V2012 使用手册

### 目录

1.IK Analyzer 2012 介绍 .....	2
2.使用指南.....	5
3.词表扩展.....	12
4.针对 solr 的分词器应用扩展.....	16
5.关于作者.....	17

## 1.IK Analyzer 2012 介绍

IK Analyzer 是一个开源的，基于 java 语言开发的轻量级的中文分词工具包。从 2006 年 12 月推出 1.0 版开始，IKAnalyzer 已经推出了 4 个大版本。最初，它是以开源项目 Luence 为应用主体的，结合词典分词和文法分析算法的中文分词组件。从 3.0 版本开始，IK 发展为面向 Java 的公用分词组件，独立于 Lucene 项目，同时提供了对 Lucene 的默认优化实现。在 2012 版本中，IK 实现了简单的分词歧义排除算法，标志着 IK 分词器从单纯的词典分词向模拟语义分词衍化。

### 1.1 IK Analyzer 2012 结构设计



## 1.2 IK Analyzer 2012 特性

- 采用了特有的“正向迭代最细粒度切分算法”，支持细粒度和智能分词两种切分模式；
- 在系统环境：Core2 i7 3.4G 双核，4G 内存，window 7 64 位，Sun JDK 1.6\_29 64 位 普通 pc 环境测试，IK2012 具有 160 万字/秒（3000KB/S）的高速处理能力。
- 2012 版本的智能分词模式支持简单的分词排歧义处理和数量词合并输出。
- 采用了多子处理器分析模式，支持：英文字母、数字、中文词汇等分词处理，兼容韩文、日文字符
- 优化的词典存储，更小的内存占用。支持用户词典扩展定义。特别的，在 2012 版本，词典支持中文，英文，数字混合词语。

## 1.3 分词效果示例

IK Analyzer 2012 版本支持 **细粒度切分** 和 **智能切分**，以下是两种切分方式的演示样例。

### 文本原文 1:

IKAnalyzer 是一个开源的，基于 java 语言开发的轻量级的中文分词工具包。从 2006 年 12 月推出 1.0 版开始，IKAnalyzer 已经推出了 3 个大版本。

- 智能分词结果:

ikanalyzer | 是 | 一个 | 开源 | 的 | 基于 | java | 语言 | 开发 | 的 | 轻量级 | 的 | 中  
文 | 分词 | 工具包 | 从 | 2006年 | 12月 | 推出 | 1.0版 | 开始 | icanalyzer | 已经 | 推  
| 出了 | 3个 | 大 | 版本

- 最细粒度分词结果:

ikanalyzer | 是 | 一个 | 一 | 个 | 开源 | 的 | 基于 | java | 语言 | 开发 | 的 | 轻量级

| 量级 | 的 | 中文 | 分词 | 工具包 | 工具 | 包 | 从 | 2006 | 年 | 12 | 月 | 推出 | 1.0 |  
版 | 开始 | ikanalyzer | 已经 | 推出 | 出了 | 3 | 个 | 大 | 版本

## 文本原文 2:

张三说的确实在理

- 智能分词结果:

张三 | 说的 | 确实 | 在理

- 最细粒度分词结果:

张三 | 三 | 说的 | 的确 | 的 | 确实 | 实在 | 在理

## 文本原文 3

公路局正在治理解放大道路面积水问题

- 智能分词结果:

公路局 | 正在 | 治理 | 解放 | 大道 | 路面 | 积水 | 问题

- 最细粒度分词结果:

公路局 | 公路 | 路局 | 正在 | 治理 | 理解 | 解放 | 放大 | 大道 | 道路 | 路面 | 面积  
| 积水 | 问题

## 文本原文 4

据路透社报道，印度尼西亚社会事务部一官员星期二(29 日)表示，日惹市附近当地时间 27  
日晨 5 时 53 分发生的里氏 6.2 级地震已经造成至少 5427 人死亡，20000 余人受伤，近 20  
万人无家可归。

- 智能分词结果:

据 | 路透社 | 报道 | 印度尼西亚 | 社会 | 事务部 | 一 | 官员 | 星期二 | 29 日 | 表示 |  
日 | 惹 | 市 | 附近 | 当地时间 | 27 日 | 晨 | 5 时 | 53 分 | 发生 | 的 | 里氏 | 6.2 级 |

地震 | 已经 | 造成 | 至少 | 5427 人 | 死亡 | 20000 | 余人 | 受伤 | 近 | 20 | 万人 |  
无家可归

- 最细粒度分词结果:

据 | 路透社 | 路透 | 社 | 报道 | 印度尼西亚 | 印度 | 尼 | 西亚 | 社会事务 | 社会 |  
事务部 | 事务 | 部 | 一 | 官员 | 星期二 | 星期 | 二 | 29 | 日 | 表示 | 日 | 惹 | 市 |  
附近 | 当地时间 | 当地 | 时间 | 27 | 日 | 晨 | 5 | 时 | 53 | 分发 | 分 | 发生 | 发 | 生  
| 的 | 里氏 | 6.2 | 级 | 地震 | 已经 | 造成 | 至少 | 5427 | 人 | 死亡 | 20000 | 余人 |  
受伤 | 近 | 20 | 万人 | 万 | 人 | 无家可归

## 2.使用指南

### 2.1 下载地址

GoogleCode 开源项目 : <http://code.google.com/p/ik-analyzer/>

GoogleCode 下载 : <http://code.google.com/p/ik-analyzer/downloads/list>

### 2.2 与相关项目的版本兼容

IK 分词器版本	Lucene 版本	Solr 版本
3.1.3GA 及先前版	兼容 2.9.1 及先前版本	没有 solr 接口
3.1.5GA	兼容 2.9.1 及先前版本	对 solr1.3 提供接口实现
3.1.6GA	兼容 2.9.1 及先前版本	对 solr1.3、solr1.4 提供接口实现
3.2.x	兼容 Lucene2.9 及 3.0 版本 不支持 Lucene2.4 及先前	仅对 solr1.4 提供接口实现

## 2.3 安装部署

IK Analyzer 安装包包含：

1. 《IKAnalyzer 中文分词器 V2012 使用手册》( 即本文档 )
2. IKAnalyzer2012.jar ( 主 jar 包 )
3. IKAnalyzer.cfg.xml ( 分词器扩展配置文件 )
4. stopword.dic ( 停止词典 )
5. LICENSE.TXT ; NOTICE.TXT ( apache 版权申明 )

它的安装部署十分简单，将 IKAnalyzer2012.jar 部署于项目的 lib 目录中；

IKAnalyzer.cfg.xml 与 stopword.dic 文件放置在 class 根目录 ( 对于 web 项目，通常是 WEB-INF/classes 目录，同 hibernate、log4j 等配置文件相同 ) 下即可。

## 2.4 Lucene 用户快速入门

### 代码样例

---

#### IKAnalyzerDemo

```
/**
 * IK 中文分词 版本 5.0
 * IK Analyzer release 5.0
 *
 * Licensed to the Apache Software Foundation (ASF) under one or more
 * contributor license agreements. See the NOTICE file distributed with
 * this work for additional information regarding copyright ownership.
 * The ASF licenses this file to You under the Apache License, Version
```

2.0

\* (the "License"); you may not use this file except in compliance with  
\* the License. You may obtain a copy of the License at

\*

\* <http://www.apache.org/licenses/LICENSE-2.0>

\*

\* Unless required by applicable law or agreed to in writing, software  
\* distributed under the License is distributed on an "AS IS" BASIS,  
\* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
\* See the License for the specific language governing permissions and  
\* limitations under the License.

\*

\* 源代码由林良益([linliangyi2005@gmail.com](mailto:linliangyi2005@gmail.com)) 提供

\* 版权声明 2012, 乌龙茶工作室

\* provided by [Linliangyi](#) and copyright 2012 by [Oolong](#) studio

\*

\*

\*/

**package** org.wltea.analyzer.sample;

**import** java.io.IOException;

**import** org.apache.lucene.analysis.Analyzer;

**import** org.apache.lucene.document.Document;

**import** org.apache.lucene.document.Field;

**import** org.apache.lucene.index.CorruptIndexException;

**import** org.apache.lucene.index.IndexReader;

**import** org.apache.lucene.index.IndexWriter;

**import** org.apache.lucene.index.IndexWriterConfig;

**import** org.apache.lucene.index.IndexWriterConfig.OpenMode;

**import** org.apache.lucene.queryParser.ParseException;

**import** org.apache.lucene.queryParser.QueryParser;

**import** org.apache.lucene.search.IndexSearcher;

**import** org.apache.lucene.search.Query;

**import** org.apache.lucene.search.ScoreDoc;

**import** org.apache.lucene.search.TopDocs;

**import** org.apache.lucene.store.Directory;

**import** org.apache.lucene.store.LockObtainFailedException;

**import** org.apache.lucene.store.RAMDirectory;

**import** org.apache.lucene.util.Version;

**import** org.wltea.analyzer.lucene.IKAnalyzer;

/\*\*

```

* IKAnalyzer 示例
* 2012-3-2
*
* 以下是结合Lucene3.4 API的写法
*
*/
public class IKAnalyzerDemo {
    public static void main(String[] args){
        //Lucene Document的域名
        String fieldName = "text";
        //检索内容
        String text = "IK Analyzer是一个结合词典分词和文法分词的中文分词开源工
具包。它使用了全新的正向迭代最细粒度切分算法。";

        //实例化IKAnalyzer分词器
        Analyzer analyzer = new IKAnalyzer();

        Directory directory = null;
        IndexWriter iwriter = null;
        IndexReader ireader = null;
        IndexSearcher isearcher = null;
        try {
            //建立内存索引对象
            directory = new RAMDirectory();

            //配置IndexWriterConfig
            IndexWriterConfig iwConfig = new
IndexWriterConfig(Version.LUCENE_34 , analyzer);
            iwConfig.setOpenMode(OpenMode.CREATE_OR_APPEND);
            iwriter = new IndexWriter(directory , iwConfig);
            //写入索引
            Document doc = new Document();
            doc.add(new Field("ID", "10000", Field.Store.YES,
Field.Index.NOT_ANALYZED));
            doc.add(new Field(fieldName, text, Field.Store.YES,
Field.Index.ANALYZED));
            iwriter.addDocument(doc);
            iwriter.close();

            //搜索过程*****
            //实例化搜索器
            ireader = IndexReader.open(directory);

```

```

        isearcher = new IndexSearcher(ireader);

        String keyword = "中文分词工具包";
        //使用QueryParser查询分析器构造Query对象
        QueryParser qp = new QueryParser(Version.LUCENE_34,
        fieldName, analyzer);
        qp.setDefaultOperator(QueryParser.AND_OPERATOR);
        Query query = qp.parse(keyword);

        //搜索相似度最高的5条记录
        TopDocs topDocs = isearcher.search(query , 5);
        System.out.println("命中: " + topDocs.totalHits);
        //输出结果
        ScoreDoc[] scoreDocs = topDocs.scoreDocs;
        for (int i = 0; i < topDocs.totalHits; i++){
            Document targetDoc = isearcher.doc(scoreDocs[i].doc);
            System.out.println("内容: " + targetDoc.toString());
        }

    } catch (CorruptIndexException e) {
        e.printStackTrace();
    } catch (LockObtainFailedException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    } catch (ParseException e) {
        e.printStackTrace();
    } finally{
        if(ireader != null){
            try {
                ireader.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        if(directory != null){
            try {
                directory.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}

```

```
}
```

## 执行结果：

命中：1

内容：Document<stored/uncompressed,indexed,tokenized<text:IK Analyzer是一个结合词典分词和文法分词的中文分词开源工具包。它使用了全新的正向迭代最细粒度切分算法。>>

---

## 2.5 关键 API 说明

(注：本文档只列出常用的、主要的 API 说明，有可能因为版本变更而与实际代码产生不一致情况，因此最准确的说明请参看 Java API DOC )

### 类 org.wltea.analyzer.lucene.IKAnalyzer

说明：IK 分词器的主类，是 IK 分词器的 Lucene Analyzer 类实现。

该类使用方法请参考 “代码样例” 章节

#### ■ public IKAnalyzer()

说明：构造函数，默认实现最细粒度切分算法

#### ■ public IKAnalyzer(boolean useSmart)

说明：构造函数

参数 1 :boolean useSmart，当为 true 时，分词器采用智能切分；当为 false 时，分词器进行最细粒度切分。

### 类 org.wltea.analyzer.core.IKSegmenter

说明：这是 IK 分词器的核心类。它是独立于 Lucene 的 Java 分词器实现。当您需要

在 Lucene 以外的环境中单独使用 IK 中文分词 组件时，IKSegmenter 正是您要找的。

■ `public IKSegmenter(Reader input , boolean useSmart)`

说明：IK 主分词器构造函数

参数 1：Reader input，字符输入读取

参数 2：boolean useSmart，是否采用智能切分策略。true 使用智能切分，false 使用最细粒度切分。

■ `public IKSegmentation(Reader input , Configuration cfg)`

说明：IK 主分词器新构造函数

参数 1：Reader input，字符输入读取

参数 2：Configuration cfg，分词器配置。用户可以定制自己的 Configuration 类，来改变词典配置。

■ `public synchronized Lexeme next()throws IOException`

说明：读取分词器切分出的下一个语义单元，如果返回 null，表示分词器已经结束。

返回值：Lexeme 语义单元对象，即相当于 Lucene 的词元对象 Token

## 类 **org.wltea.analyzer.core.Lexeme**

说明：这是 IK 分词器的语义单元对象，相当于 Lucene 中的 Token 词元对象。由于 IK 被设计为独立于 Lucene 的 Java 分词器实现，因此它需要 Lexeme 来代表分词的结果。

■ `public int getBeginPosition()`

说明：获取语义单元的起始字符在文本中的位置

返回值：int ， 语义单元相对于文本的绝对起始位置

- public int getEndPosition()

说明：获取语义单元的结束字符的下一个位置

返回值：int ， 语义单元相对于文本的绝对终止位置的下一个字符位置

- public int getLength()

说明：获取语义单元包含字符串的长度

返回值：int ， 语义单元长度 = getEndPosition – getBeginPosition

- public String getLexemeText()

说明：获取语义单元包含字符串内容

返回值：String ， 语义单元的实际内容，即分词的结果

## 2.6 IKQueryPaser 与 IK 简单搜索表达式说明

在 IK2012 版本之前，由于分词器没有处理歧义分词的能力，才使用了 IKQueryParser 来解决搜索时的歧义冲突问题。随着 2012 版本的推出，用户已经不再需要使用 IKQueryParser 来解决这样的问题。直接使用 Lucene 的 QueryParser 即可。

## 3.词表扩展

目前,IK 分词器自带的主词典拥有 27 万左右的汉语单词量。由于作者个人的精力有限，并没有对搜集到的词库进行全范围的筛选、清理。此外，对于分词组件应用场景所涉及的领域的不同，也需要各类专业词库的支持。为此，IK 分词器提供了对词典的扩展支持。

在 IK 分词器 3.1.3 以上版本，同时提供了对用户自定义的停止词（过滤词）的扩展支持。

### 3.1 基于 API 的词典扩充

IK 分词器支持使用 API 编程模型扩充您的词典和停止词典。如果您的个性化词典是存储于数据库中，这个方式应该对您适用。API 如下：

- 类 `org.wltea.analyzer.dic.Dictionary`

说明：IK 分词器的词典对象。它负责中文词汇的加载，内存管理和匹配检索。

- `public static Dictionary initial(Configuration cfg)`

说明：初始化字典实例。字典采用单例模式，一旦初始化，实例就固定。

PS:注意该方法只能调用一次。

参数 1：Configuration cfg ， 词典路径配置

返回值：Dictionary IK 词典单例

- `public static Dictionary getSingleton()`

说明：获取初始化完毕的字典单例

返回值：Dictionary IK 词典单例

- `public void addWords(Collection<String> words)`

说明：加载用户扩展的词汇列表到 IK 的主词典中，增加分词器的可识别词语。

参数 1：Collection<String> words ， 扩展的词汇列表

返回值：无

- `public void disableWords(Collection<String> words)`

说明：屏蔽词典中的词元

参数 1：Collection<String> words ， 待删除的词列表

返回值：无

## 3.2 基于配置的词典扩充

IK 分词器还支持通过配置 IKAnalyzer.cfg.xml 文件来扩充您的专有词典以及停止词典 (过滤词典)。

### 1. 部署 IKAnalyzer.cfg.xml

IKAnalyzer.cfg.xml 部署在代码根目录下 (对于 web 项目, 通常是 WEB-INF/classes 目录) 同 hibernate、log4j 等配置文件相同。

### 2. 词典文件的编辑与部署

分词器的词典文件格式是无 BOM 的 UTF-8 编码的中文文本文件, 文件扩展名不限。词典中, 每个中文词汇独立占一行, 使用\r\n 的 DOS 方式换行。(注, 如果您不了解什么是无 BOM 的 UTF-8 格式, 请保证您的词典使用 UTF-8 存储, 并在文件的头部添加一空行)。您可以参考分词器源码 org.wltea.analyzer.dic 包下的.dic 文件。

词典文件应部署在 Java 的资源路径下, 即 ClassLoader 能够加载的路径中。(推荐同 IKAnalyzer.cfg.xml 放在一起)

### 3. IKAnalyzer.cfg.xml 文件的配置

---

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
    <comment>IK Analyzer 扩展配置</comment>
    <!--用户可以在这里配置自己的扩展字典 -->
    <entry key="ext_dict">/mydict.dic;
/com/mycompany/dic/mydict2.dic</entry>

    <!--用户可以在这里配置自己的扩展停止词字典-->
    <entry key="ext_stopwords">/ext_stopword.dic</entry>
</properties>
```

---

在配置文件中, 用户可一次配置多个词典文件。文件名使用 “;” 号分隔。文件路径为相对 java 包的起始根路径。

### 3.3 使用自定义的 **Configuration** 类改变词典配置

IK 分词器使用 `org.wltea.analyzer.cfg.Configuration` 接口来提供扩展字典的配置信息，在特定情况下，用户可以根据需要，实现该接口类，定制自己的字典配置。以下是接口方法说明：

- 类 `org.wltea.analyzer.cfg.Configuration`

- `public String getMainDictionary()`

说明：获取主字典路径，**注意该路径是相对与当前系统类加载器 `ClassLoader` 的路径。**

返回值：主字典路径字符串

- `public String getQuantifierDictionary ()`

说明：获取量词字典路径，**注意该路径是相对与当前系统类加载器 `ClassLoader` 的路径。**

返回值：量词典路径字符串

- `public List<String> getExtDictionarys() ()`

说明：获取扩展字典路径列表，可一次返回多个扩展词典。**注意该路径是相对与当前系统类加载器 `ClassLoader` 的路径。**

返回值：扩展典路径字符串列表

- `public List<String> getExtStopWordDictionarys () ()`

说明：获取停止词（`stopword`）字典路径列表，可一次返回多个停止词词典。**注意该路径是相对与当前系统类加载器 `ClassLoader` 的路径。**

返回值：停止词词典路径字符串列表

## 4.针对 solr 的分词器应用扩展

IK 分词器 3.2.0 以上版本从 API 层面提供了对 solr 中 BaseTokenizerFactory 接口的扩展实现。

### 4.1 solr 的 TokenizerFactory 接口实现

- 类 org.wltea.analyzer.solr.IKTokenizerFactory

说明：该类继承与 solr 的 BaseTokenizerFactory，是 IK 分词器对 solr 项目 BaseTokenizerFactory 接口的扩展实现。

属性：useSmart。该属性决定分词器是否采用智能切分。

### 4.2 solr1.4 配置样例

---

#### 使用 IKAnalyzer 的配置

```
<schema name="example" version="1.1">

.....

<fieldType name="text" class="solr.TextField">
  <analyzer class="org.wltea.analyzer.lucene.IKAnalyzer"/>
</fieldType>

.....
</schema>
```

#### 使用IKTokenizerFactory的配置

```
<fieldType name="text" class="solr.TextField" >
  <analyzer type="index">
    <tokenizer class="org.wltea.analyzer.solr.IKTokenizerFactory"
    useSmart = "false"/>

.....

  </analyzer>
  <analyzer type="query">
```

```
<tokenizer class="org.wltea.analyzer.solr.IKTokenizerFactory"
useSmart = "false"/>

.....

</analyzer>
</fieldType>
```

## 5.关于作者

Blog : [linliangyi2007.javaeye.com](http://linliangyi2007.javaeye.com)

Email : [linliangyi2005@gmail.com](mailto:linliangyi2005@gmail.com)

( 全文终 )