Atitit.汉字分词引擎的设计与实现 attilax 总结

[、常用分词器介绍 1](#_Toc1698)

[WhitespaceAnalyzer 1](#_Toc31652)

[SimpleAnalyzer 2](#_Toc14190)

[StopAnalyzer 2](#_Toc30230)

[StandardAnalyzer 2](#_Toc23738)

[CJKAnalyzer 2](#_Toc5863)

[SmartChineseAnalyzer 2](#_Toc23619)

[对中文支持的几个分词器切分示例 2](#_Toc13816)

[IKAnalyzer 3](#_Toc23343)

[我这里在总结一下各种中文分词的对比 3](#_Toc1395)

[1. 基本介绍： 3](#_Toc26140)

[2. 开发者及开发活跃度： 4](#_Toc24117)

[3. 用户自定义词库： 4](#_Toc15962)

[4. 速度（基于官方介绍，非自己测试） 4](#_Toc16528)

[5. 算法和代码复杂度 4](#_Toc7725)

[6. 文档 5](#_Toc13319)

[7. 其它 5](#_Toc5147)

# 、常用分词器介绍

## WhitespaceAnalyzer

仅仅是去掉了空格，没有其他任何操作，不支持中文。

## SimpleAnalyzer

讲除了字母以外的符号全部去除，并且讲所有字符变为小写，需要注意的是这个分词器同样把数据也去除了，同样不支持中文。

## StopAnalyzer

这个和SimpleAnalyzer类似，不过比他增加了一个的是，在其基础上还去除了所谓的stop words，比如the, a, this这些。这个也是不支持中文的。

## StandardAnalyzer

英文方面的处理和StopAnalyzer一样的，对中文支持，使用的是单字切割。

## CJKAnalyzer

这个支持中日韩，前三个字母也就是这三个国家的缩写。这个对于中文基本上不怎么用吧，对中文的支持很烂，它是用每两个字作为分割，分割方式个人感觉比较奇葩，我会在下面比较举例。

## SmartChineseAnalyzer

中文的分词。比较标准的中文分词，对一些搜索处理的并不是很好。

## 对中文支持的几个分词器切分示例

举例句子：“我是中国人”

StandardAnalyzer：我－是－中－国－人

CJKAnalyzer：我是－是中－中国－国人

SmartChineseAnalyzer：我－是－中国－人

可以看出，SmartChineseAnalyzer认为中国是一个词，而中国人不是一个词而是两个词。CJK的分词比较奇葩，而StandardAnalyzer的分词就是一个字一个词。

## IKAnalyzer

不得不说，中文的分词器还是需要中国人自己开发，因为只有自己才最了解自己的语言。中文分词我非常推荐使用IKAnalyzer，对于中文分词比较精准，我们这里使用的是Lucene4.9的版本，所以需要下载使用FF的版本，意思是for four才是给我们4.x用的。

# 我这里在总结一下各种中文分词的对比

下面部分参考自福林雨[博客](http://irfen.me/" \o "博客" \t "http://irfen.me/5-lucene4-9-learning-record-lucene-analysis-tokenizer/_blank)。

### 1. 基本介绍：

[paoding](http://code.google.com/p/paoding/)：Lucene中文分词“庖丁解牛” Paoding Analysis

[imdict](http://code.google.com/p/imdict-chinese-analyzer/)：imdict智能词典所采用的智能中文分词程序

[mmseg4j](http://code.google.com/p/mmseg4j/)： 用 Chih-Hao Tsai 的 MMSeg [算法](http://irfen.me/tag/%e7%ae%97%e6%b3%95" \o "算法" \t "http://irfen.me/5-lucene4-9-learning-record-lucene-analysis-tokenizer/_blank) 实现的中文分词器

[ik](http://code.google.com/p/ik-analyzer/)：采用了特有的“正向迭代最细粒度切分算法”，多子处理器分析模式

### 2. 开发者及开发活跃度：

基本上所有分词都已经好几年没有更新了。

### 3. 用户自定义词库：

paoding ：支持不限制个数的用户自定义词库，纯文本格式，一行一词，使用后台线程检测词库的更新，自动编译更新过的词库到二进制版本，并加载

imdict ：暂时不支持用户自定义词库。但 原版 ICTCLAS 支持。支持用户自定义 stop words

mmseg4j ：自带sogou词库，支持名为 wordsxxx.dic， utf8文本格式的用户自定义词库，一行一词。不支持自动检测。 -Dmmseg.dic.path

ik ： 支持api级的用户词库加载，和配置级的词库文件指定，无 BOM 的 UTF-8 编码，\r\n 分割。不支持自动检测。

### 4. 速度（基于官方介绍，非自己测试）

paoding ：在PIII 1G内存个人机器上，1秒 可准确分词 100万 汉字

imdict ：483.64 (字节/秒)，259517(汉字/秒)

mmseg4j ： complex 1200kb/s左右, simple 1900kb/s左右

ik ：具有50万字/秒的高速处理能力

### 5. 算法和代码复杂度

paoding ：[svn](http://irfen.me/tag/svn" \o "svn" \t "http://irfen.me/5-lucene4-9-learning-record-lucene-analysis-tokenizer/_blank) src 目录一共1.3M，6个properties文件，48个[java](http://irfen.me/tag/java" \o "java" \t "http://irfen.me/5-lucene4-9-learning-record-lucene-analysis-tokenizer/_blank)文件，6895 行。使用不用的 Knife 切不同类型的流，不算很复杂。

imdict ：词库 6.7M（这个词库是必须的），src 目录 152k，20个java文件，2399行。使用 ICTCLAS HHMM隐马尔科夫模型，“利用大量语料库的训练来统计汉语词汇的词频和跳转概率，从而根据这些统计结果对整个汉语句子计算最似然(likelihood)的切分”

mmseg4j ： svn src 目录一共 132k，23个java文件，2089行。MMSeg 算法 ，有点复杂。

ik ： svn src 目录一共6.6M(词典文件也在里面)，22个java文件，4217行。多子处理器分析，跟paoding类似，歧义分析算法还没有弄明白。

### 6. 文档

paoding ：几乎无。代码里有一些注释，但因为实现比较复杂，读代码还是有一些难度的。

imdict ： 几乎无。 ICTCLAS 也没有详细的文档，HHMM隐马尔科夫模型的数学性太强，不太好理解。

mmseg4j ： MMSeg 算法 是英文的，但原理比较简单。实现也比较清晰。

ik ： 有一个pdf使用手册，里面有使用示例和配置说明。

### 7. 其它

paoding ：引入隐喻，设计比较合理。search 1.0 版本就用的这个。主要优势在于原生支持词库更新检测。主要劣势为作者已经不更新甚至不维护了。

imdict ：进入了 lucene trunk，原版 ictclas 在各种评测中都有不错的表现，有坚实的理论基础，不是个人山寨。缺点为暂时不支持用户词库。

mmseg4j ： 在complex基础上实现了最多分词(max-word)，但是还不成熟，还有很多需要改进的地方。

ik ：  针对Lucene[全文检索](http://irfen.me/tag/%e5%85%a8%e6%96%87%e6%a3%80%e7%b4%a2" \o "全文检索" \t "http://irfen.me/5-lucene4-9-learning-record-lucene-analysis-tokenizer/_blank)优化的查询分析器IKQueryParser