第 38 卷 第 22 期
Vol.38 No.22

计 算 机 工 程
Computer Engineering

2012 年 11 月
November 2012

• 开发研究与设计技术 • 文章编号：1000—3428(2012)22—0279—04 文献标识码：A 中图分类号：TP391

基于 Lucene 的中文分析器分词性能比较研究

义天鹏，陈启安

(厦门大学计算机科学系，福建 厦门 361005)

摘 要：针对 Lucene 自带的中文分析器分词性能不理想并且难以选择第三方分析器的问题，研究多种基于 Lucene 的中文分析器，对语句分词、分词速度、建立索引的空间与时间、检索结果以及检索速度等方面进行比较。分析结果表明，在 Lucene 框架下，基于词典分词的 Paoding 分析器总体性能最优，Lucene 自带的一元分析器分词速度最快，imdict 与 ICTCLAS4J 分析器在算法效率上存在一定改进空间。

关键词：Lucene 框架；搜索引擎；中文分词；分析器；分词速度；索引；检索

Comparison Research of Segmentation Performance for Chinese Analyzers Based on Lucene

YI Tian-peng, CHEN Qi-an

(Department of Computer Science, Xiamen University, Xiamen 361005, China)

【**Abstract**】The segmentation performance on Chinese analyzer of Lucene is insufficient, and the third party analyzer is difficult to choose. Because of this problem, this paper introduces several kinds of support Lucene analyzer, based on the experiment, sentence segmentation, word segmentation speed, index space and time, retrieval results and speed of retrieval are compared and researched. Analysis results show that, in Lucene framework, Paoding analyzer based on dictionary segmentation has the best overall performance, one-word analyzer of Lucene has the highest segmentation speed, imdict and ICTCLAS4J analyzer have greater room for improvement on the algorithm efficiency.

【**Key words**】Lucene framework; search engine; Chinese segmentation; analyzer; segmentation speed; index; retrieval

DOI: 10.3969/j.issn.1000-3428.2012.22.070

1 概述

对于中文信息的搜索引擎，中文分词是事先可用系统的前提，没有中文分词，中文文本就无法建立高效快速的索引结构。开源 Lucene 框架在改进传统全文搜索引擎倒排索引的基础上，实现分块索引机制来提高小文件索引速度，提供的索引优化机制便于动态更新索引文件[1]，但它给出的中文分析器性能不佳，因此，出现了不少第三方分析器为其提供中文分词。现有的中文分词算法主要有基于词典的字符串匹配、基于语义理解以及基于词频概率统计，而歧义识别以及新词发现是分词中的难题。本文在实验的基础上介绍各分析器，用大量实验结果数据进行比较分析，用以表征分析器的分词性能，为 Lucene 中文分析器的选择提供参考。

2 分析器综述

2.1 Lucene 附带的中文分析器

2.1.1 StandardAnalyzer 与 ChineseAnalyzer 分析器

StandardAnalyzer 与 ChineseAnalyzer 对于中文分词相当于一元分词，得到的结果是独立的中文单字，而不是真正的词汇分割。对中文文本建立索引后的索引表为单字索引，因此，索引中无关信息较多。由于检索过程需要单字分词处理，从而增加了检索时间。StandardAnalyzer 除了基本的根据空格和符号来分割原始的文本信息外，还可以完成数字、字母、E-mail 地址、IP 地址的分析，此外支持过滤词表，用来替代 StopAnalyzer 能够完成的过滤功能，得到的结果中标点符号已经被删除。

2.1.2 CJKAnalyzer 中文分析器

CJKAnalyzer 中文分析器根据汉语中 2 个字组成一个词居多的特点，实现对中文的二元切分，即将句子中每相邻的 2 个字作为一个词，同时使用 StopFilter 过滤器以及自己添加停用词表完成过滤功能。二元切分结果存在很大的索引冗余，容易得到错误的检索结果，正确率比较低。

2.2 第三方中文分析器

2.2.1 JE 分析器

分词组件 JE 采用基于词典的正向最大匹配中文分词算法(Maximum Matching, MM)，MM 算法实现原理简单，时间复杂度低，统计结果表明，MM 算法的错误切分率为 1/169[2]。JE 的 MM 算法原理如下：对于一个字符串 S，

基金项目：航空科学基金资助项目(20085568013)

作者简介：义天鹏(1987－)，男，硕士研究生，主研方向：Web 信息处理，搜索引擎，软件工程；陈启安，教授

收稿日期：2011-12-18 **修回日期**：2012-03-19 **E-mail**：yitianpeng@126.com

按从前到后的顺序扫描，对扫描的每一个字，从词库中寻找最长的匹配[3]。例如，词库中有以下几个词：AB，ABC，ABCD，假设读入的文本为“ABCDEFG”，读到 AB 时要向前看 ABC 是不是一个词，如果 ABC 是，还要再向前看 ABCD 是不是，直到发现 ABCDE 不是词典中一个词，这样就切出一个词“ABCD”来。如果词库中没有词 AB，则字 A 单独作为一个词被切分出来。JE 有超过 22 万词的词库整理，并支持词典的动态扩展。

2.2.2 IK 分析器

开源中文分析器 IK 采用多子处理器分析模式，提供 IK_CAnalyzer 和 MIK_CAnalyzer 2 个主要类，IK_CAnalyzer 类实现以词典为基础的正反向全切分，即最细粒度切分算法，它针对一个中文句子，给出所有可能的词汇切分结果。MIK_CAnalyzer 类实现了以词典为基础的正反向最大匹配切分。IK 分析器分词结果数量随句子长度呈指数增长，时间和空间开销较大。目前，自带的默认主词典有近 27 万的词汇量，且支持基于 API 的词库扩充以及配置级的词库文件指定。

2.2.3 Paoding 分析器

开源分析器 Paoding 采用完全面向对象设计，它提供了 2 种分词模式：queryMode 和 writeMode，queryMode 将内容按最大的词进行切分，即最大的词中若是包含小的词则小词将被忽略，wirteMode 则是大词、小词都会切分出来，默认采用 writeMode。Paoding 分析器支持不限制个数的用户自定义词典[4]，词典名称不限，词典所在目录不限，只要是纯文本格式、一行一词且文件以 dic 作为扩展名均可以作为词典，程序编译时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。

2.2.4 mmseg4j 分析器

mmseg4j 是用 TsaiChih-Hao 的 MMSeg 算法实现的中文分析器，它提供 Simple、Complex 分词方法，都是基于正向最大匹配算法[5]，1.6 版本开始在 Complex 分词算法基础上实现了最多分词(max-word)方法。对于 Lucene 来说，对应 Analyzer 为 SimpleAnalyzer、ComplexAnalyzer、MaxWordAnalyzer，这 3 个 Analyzer 都继承自 MMSegAnalyzer，而 MMSegAnalyzer 默认使用 max-word 方式分词。mmseg4j 自带的主词典是将近 15 万的 sogou 词库，除此之外它支持名为 wordsxxx.dic，文本文件规则为一行一词，UTF-8 格式的用户自定义词库。

2.2.5 imdict 分析器

智能分析器 imdict 是 ICTCLAS 中文分词程序基于 Java 的重新实现。它的原理是基于自然语言处理领域的隐马尔科夫模型(Hidden Markov Model, HMM)，即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。imdict 直接带有 Lucene 接口，它在 Lucene 中的分析器为 SmartChineseAnalyzer，智能分词需要词典来保存词汇的统计值，而 SmartChineseAnalyzer 内置有词典库、默认停止词库，已经经过封装，可以直接使用，它暂时不支持用户自定义词库。相对于 ICTCLAS 分词系统，imdict 对一些不必要的功能进行了删减，例如词性标注、人名识别、时间识别，得到的索引文件也较小。只是分词过程涉及大量计算，导致速度慢于基于词典的分析器。

2.2.6 ICTCLAS 与 ICTCLAS4J 分析器

采用层叠隐马尔科夫模型的 ICTCLAS(Institute of Computing Technology, Chinese Lexical Analysis System)汉语词法分析系统，主要功能包括中文分词、词性标注、命名实体识别、新词识别，同时支持用户词典。当用 C++开发搜索引擎时，可以直接调用 ICTCLAS 分词；当用 Java 开发时，则需要在静态初始化段中加载它，使用 JNI 本地接口调用 ICTCLAS.DLL，实现 Java 环境下的分词。

ICTCLAS4J 中文分词系统是在 FreeICTCLAS 基础上完成的一个 Java 开源分词项目，简化了原分词程序的复杂度。只是词典加载速度较慢，且解析器结构与 Lucene 不相同，Lucene 解析器采用流式结构处理，即字符流流过解析器，然后不断从中取出词，而 ICTCLAS4J 是基于字符串处理，处理过程为先生成分词图表进行初步分词，然后进行优化，最后进行词性标注，只是算法效率较低，处理时间较慢。

3 分析器比较

实验环境为 Windows XP 操作系统，AMD 4000+处理器 2 GB 内存。

3.1 句子分词示例结果比较

测试句子：“永和服装股份公司”，含有歧义词{和服}。分词结果比较如表 1 所示。

表 1 分词结果比较

分析器	分词结果
StandardAnalyzer、ChineseAnalyzer	{永}{和}{服}{装}{有}{限}{公}{司}
CJKAnalyzer	{永和}{和服}{服装}{装股}{股份}{份公}{公司}
JE	{永和}{服装}{股份公司}
IK_CAnalyzer	{永和}{和服}{服装}{股份公司}{股份}{公司}
MIK_CAnalyzer	{永和}{和服}{服装}{股份公司}
Paoding	{永和}{和服}{服装}{股份}{公司}
mmseg4j(Simple)	{永}{和服}{装}{股份公司}
mmseg4j(Complex)	{永}{和}{服装}{股份公司}
mmseg4j(Maxword)	{永}{和}{服装}{股份}{公司}
imdict	{永}{和}{服装}{股份公司}
ICTCLAS、ICTCLAS4J	{永/nr}{和/c}{服装/n}{股份公司/l}

从表 1 可以看出，StandardAnalyzer 和 ChineseAnalyzer 采用一元切分，CJKAnalyzer 采用二元切分，而基于词典的 JE、IK 与 Paoding 分析器都能正确得到{永和}这一词，mmseg4j 3 种分词方式结果各不相同，imdict 与 ICTCLAS/ICTCLAS4J 因为核心算法相同所以分词结果一样，但 ICTCLAS/ICTCLAS4J 分词后带有词性标注，imdict 则对此功能进行了删除。

3.2　分词速度比较

如表 2 所示，前 2 次是分别对空文本串、6.01 MB 的文本进行分词得到的时间，第 3 次是对 10 000 个包括小说、新闻、文案共 520 MB 的文本文件进行分词得到的时间。从结果可以看出，除了分析器初始化需要时间外，JE、IK、Paoding、mmseg4j 因为需要加载词典，所以空串的分词也需要一定时间。而 ICTCLAS4J 加载词典速度慢是因为它在读取词典文件时都是以若干字节为单位进行读取，这样浪费了 I/O 的带宽，增加了 I/O 的次数，这可以通过利用加缓冲的方式提高读入词典的速度。通过分析分词时间可以看出，Lucene 自带的机械分析器分词速度最快，第三方分析器中 mmseg4j 和 Paoding 的分词速度相当快，IK 因为采用全切分算法，速度略慢于它们，而采用隐马尔科夫模型的 imdict 分析器因为涉及大量计算导致分词速度较慢。ICTCLAS4J 的分词时间是 imdict 的 35 倍多，这与文献[6]给出的效率对比结果一致。Java 环境下通过 JNI 方式调用 ICTCLAS 没有体现出它在 C++环境下分词速度的优越性。由此可知，因为基于词典的分析器因为实现算法简单，时间复杂度低于基于隐马尔科夫模型的 imdict 和 ICTCLAS，所以时间上花费较少。

表 2　分词时间比较　　ms

分析器	分词时间		
	空文本串	文本文件为 6.01 MB	文本文件为 520 MB
StandardAnalyzer	15	1 361	114 843
ChineseAnalyzer	16	1 300	103 484
CJKAnalyzer	15	2 470	217 984
JE	922	8 579	610 578
IK_CAnalyzer	1 000	8 548	634 046
MIK_CAnalyzer	1 000	9 407	706 829
Paoding	9 54	5 100	401 406
mmseg4j(Simple)	1 203	3 470	217 109
mmseg4j(Complex)	1 250	5 048	348 938
mmseg4j(Maxword)	1 218	5 517	373 563
imdict	3 13	27 375	2 395 141
ICTCLAS4J	1 656	958 125	84 093 400
ICTCLAS	78	15 078	1289 000

3.3　建立索引比较

该实验是对 10 000 个包括小说、新闻、文案在内的文本文件共计 520 MB 的数据对其文件名、文件路径和文件内容建立索引得到的实验结果。在实验中，Lucene 的 IndexWrite 类 mergeFactor 大小及 maxBufferDoc 参数大小对索引的建立时间有较大影响，mergeFactor 的值较小，生成索引时速度慢；mergeFactor 的值较大，生成索引的速度快[7]。

另一参数 maxBuffereDoc 用于控制内存中存放的文档数[8]，当到达设定的 maxBufferDoc 值时进行写磁盘操作，当其值设为较大的值时，需要更多的内存，设为较小的值则会发生频繁的 I/O 操作。在实验中，当 maxBufferDoc 取 1 000，Java 虚拟机最大内存只设置为 64 MB 时 Paoding 分析器会出现内存不足。综合考虑内存需求与索引建立时间，该实验设置 mergeFactor=100、maxBufferDoc=100，在同样的参数和条件下使用各自中文分析器建立索引。由于 ICTCLAS4J 分词速度很慢，而 imdict 是 ICTCLAS 程序基于 Java 的重新实现，且它直接带有 Lucene 接口。所以，这里不使用 ICTCLAS4J 及 ICTCLAS 进行分词建立索引。

建立索引占用的空间与时间比较如表 3 所示。

表 3　建立索引占用的空间与时间比较

分析器	空间/MB	时间/ms
StandardAnalyzer	137.0	85 469
ChineseAnalyzer	138.0	113 141
CJKAnalyzer	237.0	386 641
JE	182.0	548 609
IK_CAnalyzer	208.0	666 359
MIK_CAnalyzer	204.0	799 578
Paoding	192.0	486 985
mmseg4j(Simple)	28.8	163 500
mmseg4j(Complex)	28.8	176 047
mmseg4j(Maxword)	28.8	180 250
imdict	148.0	1 159 766

由实验结果可以看出，Lucene 自带的单字分析器 StandardAnalyze 和 ChineseAnalyzer 生成索引时间和空间都是最小的，而第三方分析器中 mmseg4j 建立索引的时间和空间最小，这是因为它分词后的词语冗余最少，近乎为把原句在若干个位置进行切分。

IK 因为采用全切分算法得到词汇较多，建立索引的空间最大。imdict 在建索引时对内容分词花费的时间最多，因此在这里所需时间最多。

3.4　检索结果比较

定义　查全率=检索出的文档数量/检索系统中文档总数量。

为了得到最大的查全率，除 IK 外，检索过程对检索词解析采用与建立索引对应相同的分析器，而 IK 建立索引时采用 IK_CAnalyzer 类分词，检索时采用 MIK_CAnalyzer 类解析检索词。

该实验是用 10 000 个常用词分别对各自分析器建立索引进行检索得到数据，这 10 000 个词采用随机算法从 15 万的常用词汇与短语中随机取得。检索到的文件数目为对这 10 000 个词分别进行检索得到的文件数目总和，检索时间为得到这些文件所需时间总和。

由检索结果(表 4)可知，基于词典的分析器在检索的查全率方面明显优于 Lucene 自带的基于机械分词的分析器，而检索时间也明显小于它们，这一点与文献[9]得到的结果一致。mmseg4j 分析器虽然检索时间小很多，但检索到的结果也最少。综合考虑检索结果数和所需时间，JE 和 Paoding 分析器效果较好。

表 4 检索结果比较

分析器	检索到的文件数	检索时间/ms
StandardAnalyzer	1 212 116	143 109
ChineseAnalyzer	1 214 213	140 328
CJKAnalyzer	1 290 553	1 348 129
JE	2 011 852	75 547
IK	1 922 332	76 000
Paoding	2 034 813	96 360
mmseg4j(Simple)	9 146	2 969
mmseg4j(Complex)	9 100	2 734
mmseg4j(Maxword)	10 083	4 203
imdict	1 184 054	106 625

4 结束语

本文介绍 Lucene 自带的中文分析器，同时探讨了为 Lucene 提供支持的第三方中文分析器，并用大量实验数据对这些分析器分词性能进行比较分析。综合考虑时间、空间以及查全率，Paoding 分析器总体性能最优，而且它支持不限数量简易添加的用户自定义词库，但新兴词汇出现后才能收录进词库，这在一定程度上影响到词语的歧义识别，因此若是它结合语义理解与概率统计方法则会更优，如何设计高效的算法在保证分词速度的条件下提高新词发现与歧义识别性能是下一步研究的问题。JE 分析器虽然检索结果略少于 Paoding 分析器，但得到的索引文件占据空间比 Paoding 小，内存占用也比 Paoding 少，因此，在内存较紧张的情况下可以选择 JE 分析器。IK 虽然检索结果接近于 JE 和 Paoding 分析器，但得到的索引文件占据较大空间。而带有 Lucene 接口的智能分析器 imdict 虽然得到的索引文件较小，但建立索引花费的时间太多，得到的检索结果却较少，还有待进一步提高算法的效率。

参考文献

[1] 王学松. Lucene+nutch 搜索引擎开发[M]. 北京: 人民邮电出版社, 2008.

[2] 赵 杰. 搜索引擎技术[M]. 哈尔滨: 哈尔滨工程大学出版社, 2007.

[3] 闻玉彪, 贾时银, 邓世昆, 等. 一种改进的最大匹配中文分词算法[J]. 计算机技术与发展, 2011, 21(10): 92-94.

[4] 当前几个主要的 Lucene 中文分词器的比较[EB/OL]. (2009-08-08). http://www. iteye.com/news/9637.

[5] Tsai Chih-Hao. MMSEG: A Word Identification System for Mandarin Chinese Text Based on Two Variants of the Maximum Matching Algorithm[EB/OL]. (2000-03-12). http://technology.chtsai.org/mmseg.

[6] Google Projec Hosting. imdict-chinses-analyzer 智能词典所采用的智能中文分词程序[EB/OL]. (2011-01-15). http://code.google.com/p/imdict-chinese-analyzer/.

[7] 吴众欣, 沈家立. Lucene 分析与应用[M]. 北京: 机械工业出版社, 2008.

[8] Gospodnetic O, Hatcher E. Lucene in Action[M]. Greenwich, UK: Manning Press, 2004.

[9] 胡长春. 面向搜索引擎 Lucene 的中文分析器[J]. 计算机工程与应用, 2009, 45(12): 157-159.

编辑 陆燕菲

(上接第 278 页)

5 结束语

本文提出一种基于改进视觉词袋模型的图像标注方法，通过实验验证了 Msso-BoVW 模型的正确性和在图像标注应用中的有效性。Msso-BoVW 将多尺度变换与多核学习方法相结合，充分利用各尺度视觉特征的区别力，提升了图像特征表示的区别力，进一步提高了图像标注的准确性。但本文在实验中进行尺度变换的尺度参数是采用固定步长进行的，这对具体的图像集不一定是最优的。因此，采用自适应尺度变换步长方案和引入非线性尺度空间分析是下一步的研究内容之一。在尺度变换的同时考虑与尺度变换相适应的视觉词量化与计权方案，及引入多模态视觉特征是对 BoVW 改进研究的重要方向。

参考文献

[1] Sivic J. Video Google: A Text Retrieval Approach to Object Matching in Videos[C]//Proc. of the International Conf. on Computer Vision. Nice, France: IEEE Press, 2003.

[2] 程 蕾, 吴秀清. 局部几何特征用于目标识别[J]. 计算机工程与应用, 2010, 46(26): 191-193.

[3] López-Sastre R J, Tuytelaars T, Acevedo-Rodríguez F J, et al. Towards a More Discriminative and Semantic Visual Vocabulary[J]. Computer Vision and Image Understanding, 2010, 115(3): 415-425.

[4] Elsayad I, Martinet J, Urruty T, et al. A New Spatial Weighting Scheme for Bag-of-visual-words[C]//Proc. of IEEE International Workshop on Content-Based Multimedia Indexing. Grenoble, France: IEEE Press, 2010.

[5] Ding Guiguang, Wang Jianmin, Qin Kai. A Visual Word Weighting Scheme Based on Emerging Itemset for Video Annotatio[J]. Information Processing Letters, 2010, 110(16): 692-696.

[6] Sonnenburg S. Large Scale Multiple Kernel Learning[J]. Journal of Machine Learning Research, 2006, 7(1): 1531-1565.

[7] Chang Chih-Chung, Lin Chih-Jen. LIBSVM: A Library for Support Vector Machines[EB/OL]. (2011-11-05). http://www.csie.ntu.edu.tw/~cjlin/.

[8] van Gemert J C, Veenman C J, Smeulders A W M. Visual Word Ambiguity[J]. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2009, 32(7): 1271-1283.

[9] Yang Jun, Jiang Yugang, Hauptmann A G, et al. Evaluating Bag-of-Visual-Words Representations in Scene Classification[C]//Proc. of ACM SIGMM International Workshop on Multimedia Information Retrieval. New York, USA: ACM Press, 2007.

编辑 顾姣健