链滴

11 大 Java 开源中文分词器的使用方法和分词效果对比

作者：beejson

原文链接：https://ld246.com/article/1512377443796

来源网站：链滴

本文的目标有两个：

1、学会使用11大Java开源中文分词器

2、对比分析11大Java开源中文分词器的分词效果

本文给出了11大Java开源中文分词的使用方法以及分词结果对比代码，至于效果哪个好，那要用的人合自己的应用场景自己来判断。

11大Java开源中文分词器，不同的分词器有不同的用法，定义的接口也不一样，我们先定义一个统一接口：

```
/**
 * 获取文本的所有分词结果, 对比不同分词器结果
 */
public interface WordSegmenter {
    /**
     * 获取文本的所有分词结果
     * @param text 文本
     * @return 所有的分词结果，去除重复
     */
    default public Set<String> seg(String text) {
        return segMore(text).values().stream().collect(Collectors.toSet());
    }
    /**
     * 获取文本的所有分词结果
     * @param text 文本
     * @return 所有的分词结果，KEY 为分词器模式，VALUE 为分词器结果
     */
    public Map<String, String> segMore(String text);
}
```

从上面的定义我们知道，在Java中，同样的方法名称和参数，但是返回值不同，这种情况不可以使用载。

这两个方法的区别在于返回值，每一个分词器都可能有多种分词模式，每种模式的分词结果都可能不同，第一个方法忽略分词器模式，返回所有模式的所有不重复分词结果，第二个方法返回每一种分词模式及其对应的分词结果。

在这里，需要注意的是我们使用了Java8中的新特性默认方法，并使用stream把一个map的value转为不重复的集合。

下面我们利用这11大分词器来实现这个接口：

1、word分词器

```
@Override
public Map<String, String> segMore(String text) {
    Map<String, String> map = new HashMap<>();
    for(SegmentationAlgorithm segmentationAlgorithm : SegmentationAlgorithm.values()){
        map.put(segmentationAlgorithm.getDes(), seg(text, segmentationAlgorithm));
    }
    return map;
}
```

```
private static String seg(String text, SegmentationAlgorithm segmentationAlgorithm) {
    StringBuilder result = new StringBuilder();
    for(Word word : WordSegmenter.segWithStopWords(text, segmentationAlgorithm)){
        result.append(word.getText()).append(" ");
    }
    return result.toString();
}
```

2、Ansj分词器

```
@Override
public Map<String, String> segMore(String text) {
    Map<String, String> map = new HashMap<>();

    StringBuilder result = new StringBuilder();
    for(Term term : BaseAnalysis.parse(text)){
        result.append(term.getName()).append(" ");
    }
    map.put("BaseAnalysis", result.toString());

    result.setLength(0);
    for(Term term : ToAnalysis.parse(text)){
        result.append(term.getName()).append(" ");
    }
    map.put("ToAnalysis", result.toString());

    result.setLength(0);
    for(Term term : NlpAnalysis.parse(text)){
        result.append(term.getName()).append(" ");
    }
    map.put("NlpAnalysis", result.toString());

    result.setLength(0);
    for(Term term : IndexAnalysis.parse(text)){
        result.append(term.getName()).append(" ");
    }
    map.put("IndexAnalysis", result.toString());

    return map;
}
```

3、Stanford分词器

```
private static final StanfordCoreNLP CTB = new StanfordCoreNLP("StanfordCoreNLP-chinese-
tb");
private static final StanfordCoreNLP PKU = new StanfordCoreNLP("StanfordCoreNLP-chinese
pku");
private static final PrintStream NULL_PRINT_STREAM = new PrintStream(new NullOutputStre
m(), false);
public Map<String, String> segMore(String text) {
    Map<String, String> map = new HashMap<>();
    map.put("Stanford Beijing University segmentation", seg(PKU, text));
    map.put("Stanford Chinese Treebank segmentation", seg(CTB, text));
    return map;
```

```
}
private static String seg(StanfordCoreNLP stanfordCoreNLP, String text){
    PrintStream err = System.err;
    System.setErr(NULL_PRINT_STREAM);
    Annotation document = new Annotation(text);
    stanfordCoreNLP.annotate(document);
    List<CoreMap> sentences = document.get(CoreAnnotations.SentencesAnnotation.class);
    StringBuilder result = new StringBuilder();
    for(CoreMap sentence: sentences) {
        for (CoreLabel token: sentence.get(CoreAnnotations.TokensAnnotation.class)) {
            String word = token.get(CoreAnnotations.TextAnnotation.class);;
            result.append(word).append(" ");
        }
    }
    System.setErr(err);
    return result.toString();
}
```

4、FudanNLP分词器

```
private static CWSTagger tagger = null;
static{
    try{
        tagger = new CWSTagger("lib/fudannlp_seg.m");
        tagger.setEnFilter(true);
    }catch(Exception e){
        e.printStackTrace();
    }
}
@Override
public Map<String, String> segMore(String text) {
    Map<String, String> map = new HashMap<>();
    map.put("FudanNLP", tagger.tag(text));
    return map;
}
```

5、Jieba分词器

```
private static final JiebaSegmenter JIEBA_SEGMENTER = new JiebaSegmenter();
@Override
public Map<String, String> segMore(String text) {
    Map<String, String> map = new HashMap<>();
    map.put("INDEX", seg(text, SegMode.INDEX));
    map.put("SEARCH", seg(text, SegMode.SEARCH));
    return map;
}
private static String seg(String text, SegMode segMode) {
    StringBuilder result = new StringBuilder();
    for(SegToken token : JIEBA_SEGMENTER.process(text, segMode)){
        result.append(token.word.getToken()).append(" ");
    }
    return result.toString();
}
```

6、Jcseg分词器

```
private static final JcsegTaskConfig CONFIG = new JcsegTaskConfig();
private static final ADictionary DIC = DictionaryFactory.createDefaultDictionary(CONFIG);
static {
    CONFIG.setLoadCJKSyn(false);
    CONFIG.setLoadCJKPinyin(false);
}
@Override
public Map<String, String> segMore(String text) {
    Map<String, String> map = new HashMap<>();

    map.put("复杂模式", segText(text, JcsegTaskConfig.COMPLEX_MODE));
    map.put("简易模式", segText(text, JcsegTaskConfig.SIMPLE_MODE));

    return map;
}
private String segText(String text, int segMode) {
    StringBuilder result = new StringBuilder();
    try {
        ISegment seg = SegmentFactory.createJcseg(segMode, new Object[]{new StringReader(t
xt), CONFIG, DIC});
        IWord word = null;
        while((word=seg.next())!=null) {
            result.append(word.getValue()).append(" ");
        }
    } catch (Exception ex) {
        throw new RuntimeException(ex);
    }
    return result.toString();
}
```

7、MMSeg4j分词器

```
private static final Dictionary DIC = Dictionary.getInstance();
private static final SimpleSeg SIMPLE_SEG = new SimpleSeg(DIC);
private static final ComplexSeg COMPLEX_SEG = new ComplexSeg(DIC);
private static final MaxWordSeg MAX_WORD_SEG = new MaxWordSeg(DIC);
@Override
public Map<String, String> segMore(String text) {
    Map<String, String> map = new HashMap<>();
    map.put(SIMPLE_SEG.getClass().getSimpleName(), segText(text, SIMPLE_SEG));
    map.put(COMPLEX_SEG.getClass().getSimpleName(), segText(text, COMPLEX_SEG));
    map.put(MAX_WORD_SEG.getClass().getSimpleName(), segText(text, MAX_WORD_SEG));
    return map;
}
private String segText(String text, Seg seg) {
    StringBuilder result = new StringBuilder();
    MMSeg mmSeg = new MMSeg(new StringReader(text), seg);
    try {
        Word word = null;
        while((word=mmSeg.next())!=null) {
            result.append(word.getString()).append(" ");
        }
    } catch (IOException ex) {
        throw new RuntimeException(ex);
```

```
    }
    return result.toString();
}
```

8、IKAnalyzer分词器

```
@Override
public Map<String, String> segMore(String text) {
    Map<String, String> map = new HashMap<>();

    map.put("智能切分", segText(text, true));
    map.put("细粒度切分", segText(text, false));

    return map;
}
private String segText(String text, boolean useSmart) {
    StringBuilder result = new StringBuilder();
    IKSegmenter ik = new IKSegmenter(new StringReader(text), useSmart);
    try {
        Lexeme word = null;
        while((word=ik.next())!=null) {
            result.append(word.getLexemeText()).append(" ");
        }
    } catch (IOException ex) {
        throw new RuntimeException(ex);
    }
    return result.toString();
}
```

9、Paoding分词器

```
private static final PaodingAnalyzer ANALYZER = new PaodingAnalyzer();
@Override
public Map<String, String> segMore(String text) {
    Map<String, String> map = new HashMap<>();

    map.put("MOST_WORDS_MODE", seg(text, PaodingAnalyzer.MOST_WORDS_MODE));
    map.put("MAX_WORD_LENGTH_MODE", seg(text, PaodingAnalyzer.MAX_WORD_LENGTH
MODE));

    return map;
}
private static String seg(String text, int mode){
    ANALYZER.setMode(mode);
    StringBuilder result = new StringBuilder();
    try {
        Token reusableToken = new Token();
        TokenStream stream = ANALYZER.tokenStream("", new StringReader(text));
        Token token = null;
        while((token = stream.next(reusableToken)) != null){
            result.append(token.term()).append(" ");
        }
    } catch (Exception ex) {
        throw new RuntimeException(ex);
```

```
    }
    return result.toString();
}
```

10、smartcn分词器

```
private static final SmartChineseAnalyzer SMART_CHINESE_ANALYZER = new SmartChineseA
alyzer();
@Override
public Map<String, String> segMore(String text) {
    Map<String, String> map = new HashMap<>();
    map.put("smartcn", segText(text));
    return map;
}
private static String segText(String text) {
    StringBuilder result = new StringBuilder();
    try {
        TokenStream tokenStream = SMART_CHINESE_ANALYZER.tokenStream("text", new Strin
Reader(text));
        tokenStream.reset();
        while (tokenStream.incrementToken()){
            CharTermAttribute charTermAttribute = tokenStream.getAttribute(CharTermAttribute.
lass);
            result.append(charTermAttribute.toString()).append(" ");
        }
        tokenStream.close();
    }catch (Exception e){
        e.printStackTrace();
    }
    return result.toString();
}
```

11、HanLP分词器

```
private static final Segment N_SHORT_SEGMENT = new NShortSegment().enableCustomDicti
nary(false).enablePlaceRecognize(true).enableOrganizationRecognize(true);
private static final Segment DIJKSTRA_SEGMENT = new DijkstraSegment().enableCustomDicti
nary(false).enablePlaceRecognize(true).enableOrganizationRecognize(true);
@Override
public Map<String, String> segMore(String text) {
    Map<String, String> map = new HashMap<>();
    map.put("标准分词", standard(text));
    map.put("NLP分词", nlp(text));
    map.put("索引分词", index(text));
    map.put("N-最短路径分词", nShort(text));
    map.put("最短路径分词", shortest(text));
    map.put("极速词典分词", speed(text));
    return map;
}
private static String standard(String text) {
    StringBuilder result = new StringBuilder();
    StandardTokenizer.segment(text).forEach(term->result.append(term.word).append(" "));
    return result.toString();
}
```

```
private static String nlp(String text) {
    StringBuilder result = new StringBuilder();
    NLPTokenizer.segment(text).forEach(term->result.append(term.word).append(" "));
    return result.toString();
}
private static String index(String text) {
    StringBuilder result = new StringBuilder();
    IndexTokenizer.segment(text).forEach(term->result.append(term.word).append(" "));
    return result.toString();
}
private static String speed(String text) {
    StringBuilder result = new StringBuilder();
    SpeedTokenizer.segment(text).forEach(term->result.append(term.word).append(" "));
    return result.toString();
}
private static String nShort(String text) {
    StringBuilder result = new StringBuilder();
    N_SHORT_SEGMENT.seg(text).forEach(term->result.append(term.word).append(" "));
    return result.toString();
}
private static String shortest(String text) {
    StringBuilder result = new StringBuilder();
    DIJKSTRA_SEGMENT.seg(text).forEach(term->result.append(term.word).append(" "));
    return result.toString();
}
```

现在我们已经实现了本文的第一个目的：学会使用11大Java开源中文分词器。

最后我们来实现本文的第二个目的：对比分析11大Java开源中文分词器的分词效果，程序如下：

```
public static Map<String, Set<String>> contrast(String text){
    Map<String, Set<String>> map = new LinkedHashMap<>();
    map.put("word分词器", new WordEvaluation().seg(text));
    map.put("Stanford分词器", new StanfordEvaluation().seg(text));
    map.put("Ansj分词器", new AnsjEvaluation().seg(text));
    map.put("HanLP分词器", new HanLPEvaluation().seg(text));
    map.put("FudanNLP分词器", new FudanNLPEvaluation().seg(text));
    map.put("Jieba分词器", new JiebaEvaluation().seg(text));
    map.put("Jcseg分词器", new JcsegEvaluation().seg(text));
    map.put("MMSeg4j分词器", new MMSeg4jEvaluation().seg(text));
    map.put("IKAnalyzer分词器", new IKAnalyzerEvaluation().seg(text));
    map.put("smartcn分词器", new SmartCNEvaluation().seg(text));
    return map;
}
public static Map<String, Map<String, String>> contrastMore(String text){
    Map<String, Map<String, String>> map = new LinkedHashMap<>();
    map.put("word分词器", new WordEvaluation().segMore(text));
    map.put("Stanford分词器", new StanfordEvaluation().segMore(text));
    map.put("Ansj分词器", new AnsjEvaluation().segMore(text));
    map.put("HanLP分词器", new HanLPEvaluation().segMore(text));
    map.put("FudanNLP分词器", new FudanNLPEvaluation().segMore(text));
    map.put("Jieba分词器", new JiebaEvaluation().segMore(text));
    map.put("Jcseg分词器", new JcsegEvaluation().segMore(text));
    map.put("MMSeg4j分词器", new MMSeg4jEvaluation().segMore(text));
```

```
    map.put("IKAnalyzer分词器", new IKAnalyzerEvaluation().segMore(text));
    map.put("smartcn分词器", new SmartCNEvaluation().segMore(text));
    return map;
}
public static void show(Map<String, Set<String>> map){
    map.keySet().forEach(k -> {
        System.out.println(k + " 的分词结果：");
        AtomicInteger i = new AtomicInteger();
        map.get(k).forEach(v -> {
            System.out.println("\t" + i.incrementAndGet() + " 、" + v);
        });
    });
}
public static void showMore(Map<String, Map<String, String>> map){
    map.keySet().forEach(k->{
        System.out.println(k + " 的分词结果：");
        AtomicInteger i = new AtomicInteger();
        map.get(k).keySet().forEach(a -> {
            System.out.println("\t" + i.incrementAndGet()+ " 、【"  + a + "】\t" + map.get(k).get(
));
        });
    });
}
public static void main(String[] args) {
    show(contrast("我爱楚离陌"));
    showMore(contrastMore("我爱楚离陌"));
}
```

运行结果如下：

```
*********************************************
word分词器 的分词结果：
    1 、我 爱 楚离陌
Stanford分词器 的分词结果：
    1 、我 爱 楚 离陌
    2 、我 爱 楚离陌
Ansj分词器 的分词结果：
    1 、我 爱 楚离 陌
    2 、我 爱 楚 离 陌
HanLP分词器 的分词结果：
    1 、我 爱 楚 离 陌
smartcn分词器 的分词结果：
    1 、我 爱 楚 离 陌
FudanNLP分词器 的分词结果：
    1 、我 爱楚离陌
Jieba分词器 的分词结果：
    1 、我爱楚 离 陌
Jcseg分词器 的分词结果：
    1 、我 爱 楚 离 陌
MMSeg4j分词器 的分词结果：
    1 、我爱 楚 离 陌
IKAnalyzer分词器 的分词结果：
    1 、我 爱 楚 离 陌
*********************************************
```

```
**********************************************
word分词器 的分词结果：
    1、【全切分算法】    我 爱 楚离陌
    2、【双向最大最小匹配算法】    我 爱 楚离陌
    3、【正向最大匹配算法】    我 爱 楚离陌
    4、【双向最大匹配算法】    我 爱 楚离陌
    5、【逆向最大匹配算法】    我 爱 楚离陌
    6、【正向最小匹配算法】    我 爱 楚离陌
    7、【双向最小匹配算法】    我 爱 楚离陌
    8、【逆向最小匹配算法】    我 爱 楚离陌
Stanford分词器 的分词结果：
    1、【Stanford Chinese Treebank segmentation】    我 爱 楚离陌
    2、【Stanford Beijing University segmentation】    我 爱 楚 离陌
Ansj分词器 的分词结果：
    1、【BaseAnalysis】    我 爱 楚 离 陌
    2、【IndexAnalysis】    我 爱 楚 离 陌
    3、【ToAnalysis】    我 爱 楚 离 陌
    4、【NlpAnalysis】    我 爱 楚离 陌
HanLP分词器 的分词结果：
    1、【NLP分词】    我 爱 楚 离 陌
    2、【标准分词】    我 爱 楚 离 陌
    3、【N-最短路径分词】    我 爱 楚 离 陌
    4、【索引分词】    我 爱 楚 离 陌
    5、【最短路径分词】    我 爱 楚 离 陌
    6、【极速词典分词】    我 爱 楚 离 陌
smartcn分词器 的分词结果：
    1、【smartcn】    我 爱 楚 离 陌
FudanNLP分词器 的分词结果：
    1、【FudanNLP】    我 爱楚离陌
Jieba分词器 的分词结果：
    1、【SEARCH】    我爱楚 离 陌
    2、【INDEX】    我爱楚 离 陌
Jcseg分词器 的分词结果：
    1、【简易模式】    我 爱 楚 离 陌
    2、【复杂模式】    我 爱 楚 离 陌
MMSeg4j分词器 的分词结果：
    1、【SimpleSeg】    我爱 楚 离 陌
    2、【ComplexSeg】    我爱 楚 离 陌
    3、【MaxWordSeg】    我爱 楚 离 陌
IKAnalyzer分词器 的分词结果：
    1、【智能切分】    我 爱 楚 离 陌
    2、【细粒度切分】    我 爱 楚 离 陌
**********************************************
```

这篇文章是无意中发现的,属于转载，不喜勿喷

完成代码地址 https://github.com/ysc/cws_evaluation/blob/master/src/org/apdplat/evaluation WordSegmenter.java