词法分析

# HanLP词法分析器概览

|  |  |  |  |
| --- | --- | --- | --- |
| 任务 | 算法 | 原理 | 特点 |
| 分词 | **最短路径分词** | Viterbi | 速度和精度的最佳平衡 |
| N-最短路径分词 | 二元词网 |
| 感知机分词 | 字构词 | 精度、新词发现 |
| CRF分词 |
| 极速词典分词 | 词典分词 | 速度 |
| 词性标注 | HMM词性标注 | HMM | 速度 |
| 感知机词性标注 | 感知机 | 精度 |
| CRF词性标注 | CRF |
| 实体识别 | 中国人名识别 | HMM-Viterbi角色标注 | 速度 |
| 音译人名识别 | 层叠隐马 |
| 日本人名识别 |
| 地名识别 | HMM-Viterbi角色标注 |
| 机构名识别 | 层叠隐马 |
| 感知机命名实体识别 | 感知机 | 精度 |
| CRF命名实体识别 | CRF |

# 最短路径分词（viterbi）算法

基本思想：首先根据词典，找出字串中所有可能的词（也称全切分），然后构造词语切分有向无环图（也称作粗分词图或粗分词网）。每个词对应图中的一条有向边。若赋给相应的边长一个权值（该权值可以是常数，也可以是所构成的词的属性值），然后根据该切分图，在起点到终点的所有路径中，求出长度值（包括权值）为最短的一条路径，这条路径上包含的词就是该句子的切分结果。若每个结点处记录N个最短路径值，则该方法也称**N-最短路径算法**。

为进一步提高切分精度，在词典中增加词的属性值，即给每个词也给权重。这样每个词在汉字串中的权重不同（即构成的有向图的边不为等长）。最简单的词的权重可以用词频表示，高频词的权重大，低频词的权重小。具体的权重值可以通过大规模语料库获得。

虽然HanLP中提供了dijkstra算法的实现，但是当前HanLP中最短路径分词使用的是viterbi算法。

例子：他说的确实在理

**粗分词图的生成**



词图生成的HanLP代码为<https://github.com/hankcs/HanLP/blob/master/src/main/java/com/hankcs/hanlp/seg/WordBasedSegment.java> generateWordNet(WordNet)

**遍历计算过程和回溯分词过程**



1. node列与to列

node列的词语为粗分词网中所有的词，to列为在node列为词word\_node的情况下，后边接的所有可能的词word\_to。第1个词语前边有一个“始”词，最后一个词语后边有一个“末”词。

1. begin2node\_w的计算

表示从“始”到node词的最短路径权值。可以从待计算值所在行的node列读取出word词，在to列中以待计算值所在行开始向上查找word，找到word所在行后（以首次遇到的词为准），begin2to\_w列所对应的值就是待计算值。见图中下划线。第一个词对“始-他”的begin2node\_w的值为0。

1. node2to\_w的计算

由node+w构成的2gram串的概率，也就是转移概率，计算公式为

其中，为平滑参数值取0.1，为词语node的训练集频次值，为训练集中词形总数，为词语node和词语w的共现频次值，为平滑因子。

计算的HanLP代码为<https://github.com/hankcs/HanLP/blob/master/src/main/java/com/hankcs/hanlp/utility/MathUtility.java> calculateWeight(Vertex from, Vertex to)。“始”的频次取为，“始-他”的共现频次值为“他”作为句首的频次，“理-末”的共现频次值为“理”作为句末的频次。

1. begin2to\_w\_n的计算

表示从“始”到to词的最短路径权值。begin2to\_w\_n = begin2node\_w + node2to\_w。

1. begin2to\_w\_o

表示记录在to词下的，到to词的最短路径权值，它的初始值为0，之后由begin2to\_w来更新。

1. from

表示词语to的前驱词。

，

可以看表中(7,9),(8,10),(11,13),(12,14),(15,16),(17,18)成对行来验证该公式，其中只有(17.18)行满足了第3个式子。

1. begin2to\_w的计算

表示从“始”到to词的最短路径权值。

可以看表中(7,9),(8,10),(11,13),(12,14),(15,16),(17,18)成对行来验证该公式，其中只有(17.18)行满足了第3个式子。

（6）和（7）的HanLP实现代码<https://github.com/hankcs/HanLP/blob/master/src/main/java/com/hankcs/hanlp/seg/common/Vertex.java> updateFrom(Vertex from)

1. 回溯确定分词路径

从“末”开始向前回溯，末->理->在->确实->的->说->他，可以看表中黄色单元格进行验证。

经过(6)、（7）两步，可以确保粗分词网中任意词的前驱都是最短路径的。

遍历计算过程和回溯过程的HanLP代码<https://github.com/hankcs/HanLP/blob/master/src/main/java/com/hankcs/hanlp/seg/Viterbi/ViterbiSegment.java> viterbi(WordNet wordNet)

|  |
| --- |
| private static List<Vertex> viterbi(WordNet wordNet)  {  // 避免生成对象，优化速度  LinkedList<Vertex> nodes[] = wordNet.getVertexes();  LinkedList<Vertex> vertexList = new LinkedList<Vertex>();  for (Vertex node : nodes[1])  {  node.updateFrom(nodes[0].getFirst());  }  for (int i = 1; i < nodes.length - 1; ++i)  {  LinkedList<Vertex> nodeArray = nodes[i];  if (nodeArray == null) continue;  for (Vertex node : nodeArray)  {  if (node.from == null) continue;  for (Vertex to : nodes[i + node.realWord.length()])  {  System.out.print(node.realWord + "\t" + to.realWord + "\t");  to.updateFrom(node);  }  }  }  Vertex from = nodes[nodes.length - 1].getFirst();  while (from != null)  {  vertexList.addFirst(from);  from = from.from;  }  return vertexList;  } |

# N-最短路径分词

前边已经提到，在最短路径分词中，若每个结点处记录N个最短路径值，则该方法称为**N-最短路径算法**。在HanLP中通过两个类ViterbiSegment和NshortSegment分别实现了最短路径分词和N-最短路径分词。首先看看两个类的关系。