分词

中文分词就是对中文断句，这样能消除文字的部分歧义。除了基本的分词功能，为了消除歧义还可以进行更多的加工。中文分词可以分成如下几个子任务。

分词：把输入的标题或者文本内容等分成词。

词性标注（POS）：给分出来的词标注上名词或动词等词性。词性标注可以部分消除词的歧义，例如"行"作为量词和作为形容词表示的意思不一样。

语义标注：把每个词标注上语义编码。

很多分词方法都借助词库。词库的来源是语料库或者词典，例如"人民日报语料库"或者《现代汉语大词典》。

目前中文分词主要有两种思路：查词典和字标注。首先，查词典的方法有：机械的最大匹配法、最少词数法，以及基于有向无环图的最大概率组合，还有基于语言模型的最大概率组合，等等。查词典的方法简单高效（得益于动态规划的思想），尤其是结合了语言模型的最大概率法，能够很好地解决歧义问题，但对于中文分词一大难度——未登录词（中文分词有两大难度：歧义和未登录词），则无法解决；为此，人们也提出了基于字标注的思路，所谓字标注，就是通过几个标记（比如4标注的是：single，单字成词；begin，多字词的开头；middle，三字以上词语的中间部分；end，多字词的结尾），把句子的正确分词法表示出来。这是一个序列（输入句子）到序列（标记序列）的过程，能够较好地解决未登录词的问题，但速度较慢，而且对于已经有了完备词典的场景下，字标注的分词效果可能也不如查词典方法。总之，各有优缺点（似乎是废话～），实际使用可能会结合两者，像结巴分词，用的是有向无环图的最大概率组合，而对于连续的单字，则使用字标注的HMM模型来识别。

**中文分词流程与结构**

中文分词总体流程与结构如图4-8所示。

简化版本的中文分词切分过程说明如下。

生成全切分词图：根据基本词库对句子进行全切分，并且生成一个邻接链表表示的词图。

计算最佳切分路径：在这个词图的基础上，运用动态规划算法生成切分最佳路径。

词性标注：可以采用HMM方法进行词性标注。

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20110615/135323974.jpg> |
| 图4-8  中文分词结构图 |

未登录词识别：应用规则识别未登录词。

按需要的格式输出结果。

复杂版本的中文分词切分过程说明如下。

对输入字符串切分成句子：对一段文本进行切分，依次从这段文本中切分出一个句子，然后对这个句子再进行切分。

原子切分：对于一个句子的切分，首先是通过原子切分，将整个句子切分成一个个的原子单元（即不可再切分的形式，例如ATM这样的英文单词可以看成不可再切分的）。

生成全切分词图：根据基本词库对句子进行全切分，并且生成一个邻接链表表示的词图。

计算最佳切分路径：在这个词图的基础上，运用动态规划算法生成切分最佳路径。

未登录词识别：进行中国人名、外国人名、地名、机构名等未登录名词的识别。

重新计算最佳切分路径。

词性标注：可以采用HMM方法或最大熵方法等进行词性标注。

根据规则调整切分结果：根据每个分词的词形以及词性进行简单的规则处理，如日期分词的合并。

按需要的格式输出结果：例如输出成Lucene需要的格式。

**概率语言模型的分词方法**

从统计思想的角度来看，分词问题的输入是一个字串C=C1,C2,……,Cn，输出是一个词串S=W1,W2,……,Wm，其中m<=n。对于一个特定的字符串C，会有多个切分方案S对应，分词的任务就是在这些S中找出概率最大的一个切分方案，也就是对输入字符串切分出最有可能的词序列。

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20110615/140210949.jpg> |

例如对于输入字符串C"有意见分歧"，有S1和S2两种切分可能。

S1：有/  意见/  分歧/

S2：有意/  见/  分歧/

计算条件概率P(S1|C)和P(S2|C)，然后采用概率大的值对应的切分方案。根据贝叶斯公式，有[http://images.51cto.com/files/uploadimg/20110615/140247308.jpg](http://images.51cto.com/files/uploadimg/20110615/140247308.jpg) 。

其中P(C)是字符串在语料库中出现的概率，只是一个用来归一化的固定值。从词串恢复到汉字串的概率只有唯一的一种方式，所以P(C|S)=1。因此，比较P(S1|C)和P(S2|C)的大小变成比较P(S1)和P(S2)的大小。

概率语言模型分词的任务是：在全切分所得的所有结果中求某个切分方案S，使得P(S)最大。那么，如何来表示P(S)呢？为了容易实现，假设每个词之间的概率是上下文无关的，则：

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20110615/140339926.jpg> |

其中，对于不同的S，m的值是不一样的，一般来说m越大，P(S)会越小。也就是说，分出的词越多，概率越小。这符合实际的观察，如最大长度匹配切分往往会使得m较小。计算任意一个词出现的概率如下：

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20110615/140428691.jpg> |

因此

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20110615/140454259.jpg> |

从另外一个角度来看，计算最大概率等于求切分词图的最短路径。但是这里不采用Dijkstra算法，而采用动态规划的方法求解最短路径。

常用的词语概率表如表4-3所示。

表4-3  词语概率表



P(S1) = P(有)   P(意见)   P(分歧) = 1.8 × 10-9

P(S2) = P(有意)   P(见)   P(分歧) = 1×10-11

可得P(S1) > P(S2)，所以选择S1对应的切分。

如何尽快找到概率最大的词串？因为假设每个词之间的概率是上下文无关的，因此满足用动态规划求解所要求的最优子结构性质和无后效性。在动态规划求解的过程中并没有先生成所有可能的切分路径Si，而是求出值最大的P(Si)后，利用回溯的方法直接输出Si。

到节点Nodei为止的最大概率称为节点Nodei的概率：

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20110615/141218864.jpg> |

如果Wj的结束节点是Nodei，就称Wj为Nodei的前驱词。这里的prev（Nodei）就是节点i的前驱词集合。

比如上面的例子中，候选词"有"就是节点1的前驱词，"意见"和"见"都是节点3的前驱词。

StartNode（wj）是wj 的开始节点，也是节点i的前驱节点。

因此切分的最大概率max(P(S))就是P(Nodem)=P(节点m的最佳前驱节点) P(节点m的最佳前驱词)。

**N元分词方法**

在介绍N元模型之前，让我们先来做个香农游戏（Shannon Game）。我们给定一个词，然后猜测下一个词是什么。当我说"NBA"这个词时，你想到下一个词是什么呢？我想大家有可能会想到"篮球"，基本上不会有人想到"足球"吧。

切分出来的词序列越通顺，越有可能是正确的切分方案。N元模型主要用来衡量词序列搭配的合理性。N元模型指句子中在n个单词序列后出现的单词w的概率。

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20110615/153944553.jpg> |
|  |

但是这种方法存在两个致命缺陷：一个缺陷是参数空间过大，不可能实用化；另外一个缺陷是数据稀疏严重。为了解决这两个问题，我们引入了马尔科夫假设：一个词的出现仅仅依赖于它前面出现的有限的一个或者几个词。

如果简化成一个词的出现仅仅依赖于它前面出现的一个词，那么就称为二元模型（Bigram），即：

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20110615/154018604.jpg> |
|  |

如果简化成一个词的出现仅仅依赖于它前面出现的两个词，就称为三元模型（Trigram）。

在实践中用得最多的就是二元模型和三元模型了，而且效果很不错。高于四元的模型用得很少，因为训练它需要更庞大的语料，而且数据稀疏严重，时间复杂度高，精度却提高得不多。

二元模型考虑一个单词后出现另外一个单词的概率，是N元模型中的一种。例如：一般来说，"中国"之后出现"北京"的概率大于"中国"之后出现"北海"的概率，也就是：

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20110615/154047427.jpg> |
|  |

二元词表的格式是"左词@右词:组合频率"，例如：

1. 中国@北京:100
2. 中国@北海:1

可以把二元词表看成是基本词表的常用搭配。分词初始化时，先加载基本词表，对每个词编号，然后加载二元词表，只存储词的编号。

对于拼音转换等歧义较多的情况也可以采用三元模型（Trigram），例如：

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20110615/154248856.jpg> |
|  |

因为有些词作为开始词的可能性比较大，例如"在那遥远的地方"、"在很久以前"，这两个短语都以"在"这个词作为开始词。因此，在实际的N元分词过程中，增加虚拟的开始节点（Start）和结束节点（End），分词过程中考虑P（在|Start）。如果把"有意见分歧"当成一个完整的输入，分词结果实际是："Start/  有/  意见/   分歧/   End"。

下面我们来实现二元连接。想象在下跳棋，跳两次涉及3个位置。二元连接中的前后两个词涉及3个节点。

1. //计算节点i的最佳前驱节点
2. void getBestPrev(AdjList g,int i)   {
3. Iterator**<CnToken>** it1 = g.getPrev(i);//得到一级前驱词集合
4. double maxProb = Double.NEGATIVE\_INFINITY;
5. int maxPrev1 = -1;
6. int maxPrev2 = -1;
8. while(it1.hasNext()) {
9. CnToken t1 = it1.next();
10. //得到一级前驱词对应的二级前驱词集合
11. Iterator**<CnToken>** it2 = g.getPrev(t1.start);
12. while(it2.hasNext()) {
13. CnToken t2 = it2.next();
15. int bigramFreq=getBigramFreq  
    (t2,t1);//从二元词典找二元频率
16. //平滑后的二元概率
17. double biProb = lambda1\*t1.freq  
    + lambda2\*(bigramFreq/t2.
18. freq);
19. double nodeProb = prob[t2.start]  
    +(Math.log(biProb));
21. if (nodeProb **>** maxProb) {//概率最大的算作最佳前趋
22. maxPrev1 = t1.start;
23. maxPrev2 = t2.start;
24. maxProb = nodeProb;
25. }
26. }
27. }
28. prob[i] = maxProb;
29. prev1Node[i] = maxPrev1;
30. prev2Node[i] = maxPrev2;
31. }

查找N元词典的方法有：可以采用Trie树的形式来存放N元模型的参数。与词典Trie树的区别在于：词典Trie树上每个节点对应一个汉字，而N元模型Trie树的一个节点对应一个词。或者可以把搭配信息存放在词典Trie树的叶子节点上。存储从词编号到频率的映射，采用折半查找。

1. public class BigramMap {
2. public int[] keys;//词编号
3. public int[] vals;//频率
4. }

在自然语言处理中，N元模型可以应用于字符，衡量字符之间的搭配；或者应用于词，衡量词之间的搭配。可以应用于编码识别，将要识别的文本按照GB码和BIG5码分别识别成不同的汉字串，然后计算其中所有汉字频率的乘积，取乘积大的一种编码。

**新词发现**

词典中没有的，但是结合紧密的字或词有可能组成一个新词。 比如："水立方"如果不在词典中，可能会切分成两个词"水"和"立方"。如果在一篇文档中"水"和"立方"结合紧密，则"水立方"可能是一个新词。可以用信息熵来度量两个词的结合紧密程度。信息熵的一般公式是：

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20110615/154558308.jpg> |
|  |

如果Ｘ和Ｙ的出现相互独立，则P(Ｘ,Ｙ)的值和P(Ｘ)P(Ｙ)的值相等，I(Ｘ,Ｙ)为0。如果Ｘ和Ｙ密切相关，P(Ｘ,Ｙ)将比P(Ｘ)P(Ｙ)大很多，I(Ｘ,Ｙ)值也就远大于0。如果Ｘ和Ｙ的几乎不会相邻出现，而它们各自出现的概率又比较大，那么I (Ｘ,Ｙ)将取负值，这时候Ｘ和Ｙ负相关。设f(C)是词C出现的次数，N是文档的总词数，则：

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20110615/154624285.jpg> |
|  |

因此，两个词的信息熵计算如下：

|  |
| --- |
| <http://images.51cto.com/files/uploadimg/20110615/154653153.jpg> |

新词语有些具有普遍意义的构词规则，例如"模仿秀"由"动词+名词"组成。统计的方法和规则的方法结合对每个文档中重复子串组成的候选新词打分，超过阈值的候选新词选定为新词。此外，可以用Web信息挖掘的方法辅助发现新词：网页锚点上的文字可能是新词，例如"美甲"。另外，可以考虑先对文档集合聚类，然后从聚合出来的相关文档中挖掘新词。

**未登录词识别**

"南京市长叫江大桥？""你怎么知道的？""因为看到一个标语--南京市长江大桥欢迎您。"

未登录词在英文中叫做Out Of Vocabulary（简称OOV）词。常见的未登录词包括人名、地名、机构名。

对识别未登录词有用的信息说明如下。

未登录词所在的上下文。例如："\*\*教授"，这里"教授"是人名的下文；"邀请\*\*"，这里"邀请"是人名的上文。

未登录词本身的概率。例如：不依赖上下文，直观地来看，"刘宇"可能是个人名，"史光"不太可能是个人名。采用未登录词的概率作为这种可能性的衡量依据。"刘宇"作为人名的概率等于"刘宇"作为人名出现的次数除以人名出现的总次数。

例如：我爸叫李刚。这里"动词 + 姓 + 名 + 标点符号"组成了一个识别规则。可以根据这个识别规则识别出"李刚"这个人名。这个规则的完整形式是：

动词 + 中国人名 + 标点符号 => 动词 + 姓 + 名 + 标点符号

所以可以通过匹配规则来识别未登录词。为了实现同时查找多个规则，可以把右边的模式组织成Trie树，左边的模式作为节点属性。全切分词图匹配上右边的模式后用左边的模式替换。

可以用二元模型或三元模型来整合未登录词本身的概率和未登录词所在的上下文这两种信息。

未登录地名识别过程说明如下。

选取未登录地名候选串。

未登录地名特征识别。

对每个候选未登录地名根据特征判断是否真的地名。判断方法可以用SVM二值分类。

整合地名词图。

参考资料：

[【中文分词系列】 1. 基于AC自动机的快速分词](https://spaces.ac.cn/archives/3908)

[【中文分词系列】 2. 基于切分的新词发现](https://spaces.ac.cn/archives/3913)

[【中文分词系列】 3. 字标注法与HMM模型](https://spaces.ac.cn/archives/3922)

[【中文分词系列】 4. 基于双向LSTM的seq2seq字标注](https://spaces.ac.cn/archives/3924)

[【中文分词系列】 5. 基于语言模型的无监督分词](https://spaces.ac.cn/archives/3956)

[【中文分词系列】 6. 基于全卷积网络的中文分词](https://spaces.ac.cn/archives/4195)

[【中文分词系列】 7. 深度学习分词？只需一个词典！](https://spaces.ac.cn/archives/4245)

[【中文分词系列】 8. 更好的新词发现算法](https://spaces.ac.cn/archives/4256)