|  |
| --- |
| **几类不同的词典结构和查找算法对汉语分词系统性能的影响研究** |
|  |
| **Dingzirui Wang** |
| Harbin Institute of Technology |
| 1183710211@stu.hit.edu.cn |
|  |
|  |

摘要

中文词法分析任务是中文信息处理中的一项基本任务，主要包括两大部分，分别是中文分词和中文词性标注。而在中文分词任务中，对效率和正确性影响最大的两个因素就是词典的结构和分词算法的实现。本论文的目的是针对中文分词任务的特点，选取了一些具有代表性的数据结构和算法，尝试对这一问题进行解决，并比较了不同方法之间运行效率和正确性上的差异。

对于词典结构，在最基础的链表的基础上，考虑到单词按照词典序具有有序性这一性质，可以采用二分查找的方法来进行求解。对于查找问题，哈希算法也是一个十分简单而有效的算法。此外，在字符查找问题中，Trie树是一种非常经典的数据结构，本文实现了双数组Trie并进行分析。

然后对于查找算法，本文实现了最基础的前后向最大匹配算法，除此之外，本文还实现了全切分有向图最大匹配算法，并对此进行分析。

最后，本文对以上词典结构和查找算法的两两组合得到的结果进行分析，并研究了各个算法的不足，以及可能的优化和相关工作。

简介

在基于中文的自然语言处理中, 由于不像英文中词与词之间有固定的自然分界符, 所以对中文进行分词通常是处理中文信息任务的基础;同时由于中文自身的复杂性, 中文分词也一直是中文信息处理任务的难题。中文分词时进行词性标注、命名实体识别、关键词提取和文本聚类等下游自然语言处理任务的基础，也是语义分析等深层次文本里接任务的基础，在中文搜索引擎、输入法、机器翻译和智能问答系统等应用中起到了支撑作用。中文分词作为中文自然语言处理领域的重要基础研究, 近些年来很多专家学者致力于该领域的研究, 研究方法主要分为三种:a) 基于规则的方法;b) 基于传统机器学习模型的方法;c) 基于深度神经网络模型的方法。

基于规则的方法利用构词原理结合标注的词性等信息, 构建基于句法—语义规则的分析系统, 配合语法信息字典, 并补充了大量消除歧义的信息。Sui Z([2002](#Sui2002))在语法规则的基础上增加了领域特征进行中文分词的处理。基于规则的方法的优点是具有针对性和暂时较高的准确率, 但由于句法构造的领域相关性, 适应性较差, 词典与歧义消解处理难维护。

随着SIGHAN国际中文分词评测Bakeoff的展开, 将中文分词任务视做序列标注问题来解决逐渐成为主流。基于传统机器学习模型的方法主要为基于字标注的机器学习模型方法, 即字在字串的标注问题, 该方法能平等地看待词典词和未登录词的识别。

为了比较几个传统算法性能上的差距，本文主要完成了以下几个工作：

* 实现了多个词典结构和搜索算法。
* 在1998 年人民日报局部语料上取得了优异的结果，一个大规模的中文分词语料库。
* 实验表明传统算法虽然能取得较高的正确率，但在泛化能力和执行效率上有着极大的限制。

图1: 词频和出现次数的统计图，其中横坐标词频，纵坐标为出现次数的自然对数。

任务定义

本文研究了中文分词任务。这一任务是对一个连续的字序列按照一定的规范重新组合成词序列的过程。特别的，本文将实现的算法应用到1998 年人民日报局部语料上，这是一个可用于中文分词和词性标注的大规模语料库。首先，本文根据已分词的文本来获取词典。在获得词典后，分词系统将会根据词典来对这一语料进行分词。原文中的每一行都会被替换为一系列词，每两个词之间都会用空格来分隔。

词典分析

我们首先根据已分词的文本，去除掉词性和所有辅助标注的符号后，将每个单词看作单词表中的一个条目，并记录每个单词出现的频率。此外，出于后续测试中可扩展性的考虑，我们同时提取了一元和二元词袋。

这样得到的词典中，每一个条目为一个三元组，其中为组成词组的单词的数量；为词组出现的频率；为词表中的词组，为全体中文字符的闭包。

整个词典按照的字典序升序排布，便于词典组织结构的处理。

对词典的词频统计如图[1](#图1)所示，其中横坐标为词频，纵坐标为。可以看到绝大多数的词仅出现了一次，占到了总数的74.16%。之后出现次数和词频的对数大致呈负对数的关系，表明二者之间可能可以用一个函数关系来拟合。

最后，词典中出现频率最高的单词为”，”，出现的次数高达74921次。

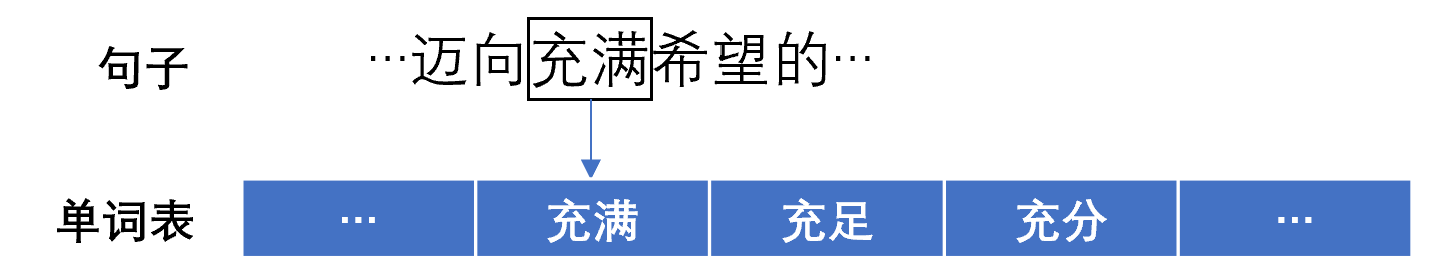


图2：分词系统示意图

分词模型

下面介绍本实验中用到的算法和模型结构。

图[2](#图2)给出了该模型的一个大体框架。算法在读取完词典文件后，会根据特定结构将所有单词组织起来。然后，算法（分词器）会逐行对文本进行分词，每次根据特定的规则提取一个可能的词组，并分析该词组是否在词典中存在，若存在，则将其作为一个分词的结果提取出来。

这一节如下组织。首先会介绍程序中各种实现的词典的结构（§4.1），之后会介绍各种分词器的结构（§4.2）。

词典结构

我们来介绍词典组织结构，其用于在给定了一个词之后，如何在单词表中检索这个词是否存在。

首先我们采用了最原始的链表查询（§4.1.1）。然后基于待查询量为字符串的这一特点，尝试了二分搜索（§4.1.2）和哈希桶（§4.1.3）这两种算法。最后，我们尝试了DAT（§4.1.4），一种预处理时间较长但检索速度极快的结构。

在下文中，为了简化叙述，我们记N为单词表的大小，L为最大的词组长度。

* + 1. 链表

由于思路简单，实现方便的特性，链表被广泛地应用在各个场景中。

对于该任务，我们每次将新读取的单词放到列表的尾部。对于每次查询，我们遍历整个列表，看待查询的词是否存在。

该结构的插入复杂度为O(1)，查询复杂度为O(N)。因此该结构的预处理性能极佳，但查询的性能非常差，对于大规模的语料库，甚至不能在可接受的时间范围内得到结果。

* + 1. 二分查找

由于单词按照字典序具有有序性这一性质，因此可以使用二分搜索来解决这一问题。

我们按照组成词组的单词数量来进行分类，对于单词表中的每一个词，我们将其放到其长度对应的数组的末尾。由于单词表已经有序，所以按先后顺序放入的词组也是有序的。

对于每次查询，我们按照词组长度由小到大的顺序，在数组中进行二分查找，直到查询到该词组，或发现该词组不存在。

由于单词表的性质，该结构的插入复杂度为O(1)，查询复杂度为O(LlogN)。

* + 1. 桶哈希

求解一个词组的哈希值分为两步。首先，求解组成词组的每个单字的哈希值；之后，根据每个单子的哈希值，求出整个词组的哈希值。

对于每个单字，由于其采用GBK编码格式，所以我们可以通过其编码，来还原出它的编号，具体来说，有如下公式：

其中和分别表示第i位的低16位和高16位的值。

之后，我们采用BKDR哈希函数来求解词组的哈希值，具体来说，有如下公式：

其中seed为哈希种子，mod一般取一个大质数。在我们的程序中，seed为131，mod为0xFFFFF。

我们会给每一个求得的哈希值创建一个数组，称为桶。产生冲突时，会在每个桶的末尾添加该元素。

对于每次查询，我们首先计算词组的哈希值，然后在哈希值对应的桶中，看该词组是否存在。

该结构的插入复杂度为O(1)，对于查询操作，我们可以认为所有的词几乎均匀地分布在各个桶中，因此查询复杂度为O(N/mod)。

* + 1. 双数组Trie

在单词检索问题中，Trie树是一个十分经典而有效的结构。而在目前Trie的实现中，Jun-ichi(1992)的双数组实现是较为有效的一种。

我们用上一节中提到的单字哈希来作为分支的依据，该结构的插入复杂度为O(L)，查询复杂度为O(L)。

分词器

这一节我们介绍分词算法，其用于在给定单词表之后，每次接收一个句子，并按照一定的逻辑来对句子进行分词。

我们主要采用了两种方法。前后向最大匹配（§4.2.1），一种基于贪心的算法；全切分有向图搜索（§4.2.2），一种基于DP的算法。

以下的讨论均针对单句的分词。

* + 1. 前后向最大匹配

这种算法基于一种非常朴素的思想，每次我们想要查找一个分割出来的词组，总是希望这个词组的长度是最大的。

对于前向最大匹配算法，我们每次固定起点，然后从后往前枚举终点，看起点和终点之间的字能否组成一个词组，若可以，则记录下来；否则，终点往前挪一次。

而对于后向最大匹配，我们固定终点，往后挪起点即可。

显然这一算法的时间复杂度为O(L^2)，实验表明对于大规模语料库，这一效率还是勉强可以接受的。

* + 1. 全切分有向图

这种算法基于的思想是，首先构造出所有可行的分词方案，然后我们给定一个指标，从所有方案中找出最满足这一指标。

我们枚举句子的所有区间，看该区间内的字能否组成一个词组，如果可以，则将其看作图中的一个点，其前驱为所有以该词起点为终点的节点，后继为所有以该词终点为起点的节点。

然后借用DP的思想，按照句子中所有词组在语料中的频率的乘积的大小来作为评价指标，这里为了防止结果过小，故采用对数和，则状态转移方程为：

其中表示j到i这个词组在语料中的频率。

在求得最后一个位置的f后，就可以按照f的值，逆着找到算法选择的路径，从而得到最终的分词结果。

虽然这一算法的理论复杂度也为O(L^2)，但由于对于一个位置，算法探测多次后面可行的词组终点，因此在实验中的时间性能表现会比前后向最大匹配差很多。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 模型 | 分词时间 | 精确率 | 召回率 | F1 |
| 链表+前向最大匹配（多进程） | >20h | - | - | - |
| 二分搜索+前向最大匹配 | >2h | - | - | - |
| 二分搜索+前向最大匹配（多进程） | 21.9min | 98.55% | 99.00% | 0.9877 |
| 二分搜索+全切分有向图（多进程） | 21.0min | 92.77% | 86.92% | 0.8975 |
| 桶哈希+前向最大匹配 | >2h | - | - | - |
| 桶哈希+前向最大匹配（多进程） | 21.6min | 98.97% | 99.47% | 0.9922 |
| 桶哈希+后向最大匹配（多进程） | 20.7min | 98.91% | 99.44% | 0.9918 |
| 桶哈希+前向最大匹配（多进程、1-gram） | 3.8min | 97.05% | 97.82% | 0.9743 |
| 桶哈希+全切分有向图（多进程） | 25.4min | 92.95% | 87.09% | 0.8992 |
| DAT+前后向 | >2h | - | - | - |
| DAT+前后向（多进程） | 20.6min | 98.63% | 99.21% | 0.9921 |
| DAT+全切分有向图（多进程） | 24.3min | 92.81% | 87.01% | 0.8988 |
| 表1：各模型在PFR上的时间性能和分词效果。实验设备为AMD Ryzen 5 3600X，开启多进程数量为9。未注明分词结果的部分由于执行时间过长，故仅统计大概的运行时间。若未注明则默认使用2-gram。 | | | | |

实验

我们将我们的模型应用到PFR，一个基于1998年人民日报语料的大规模中文分词和词性标注语料库。其中的每一个句子都已经被分好词，并且在词后添加了词性标注。

由于我们并不需要词性的信息，所以我们将已分词文件去掉词性后，按照指定的格式提取出单词表。

然后根据单词表，执行不同的词典结构和分词器的组合，并对结果进行分析评估。

最终得到的结果如表[1](#表1)所示。

* + 1. 执行时间

可以看到链表由于较差的结构性质，即使是启用多进程，运行的时间依旧长到不可接受。

而对于其他词典结构，运行时间相差不大。即使是理论速度较快的DAT，表现依然欠佳，我们认为是因为实现中过多地用到了List，抵消了结构本身带来的性能上的提升。

对于不同的分词器，可以看到全切分有向图虽然理论上时间复杂度的常数要大于最大匹配算法，但在实际运行中，和最大匹配算法的区别并不大，在可接受范围内。

此外，对于1-gram和2-gram模型，可以看到运行时间成倍地增长，表明扩大词袋会显著地影响分词效率。

* + 1. 分词效果

可以看到整体的分词准确率均在90%以上，说明各项算法的分词效果都有较高的水准。

对于同一种算法，可以看到2-gram相对1-gram，有着1%~2%的性能提升，虽然数值并不大，但考虑到2-gram已经达到了99%左右的准确率，所以这种提升还是让人比较满意的。

相关工作

上世纪八十年代初就有学者开始研究自动分词系统，并且陆续有一些实用性系统出现，典型的有CDWS（[Nanyuan, 1987](#Nanyun1987)）等。由于受硬件条件及分词技术的影响，早期分词系统在分词速度和精度上还不够理想。实用性不高。但这些分词系统的出现为后续分词系统设计打下了良好基础。

现代分词系统已经几乎都建立在基于统计和深度学习的基础之上，最典型的代表就是中国科学院计算所汉语词法分析系统，其能够达到98.45%的分词精度，词典数据压缩后不到3M（[ICT, 2010](#ICT2010)）。

除了上述大规模的分词系统外，还有很多更常用的中文分词API库，如Jieba等，极大的方便了中文分词的相关研究。

结论

对于各种传统算法，准确率都能保持在90%以上。因此，分词算法的提升应当更注重泛化性或者针对某个特定领域。

运行时间上，各个传统算法的表现都差强人意，万句级别的语料库就已经需要几十分钟的运行时间。因此，如何提升分词算法的执行速度也是一个需要关注的问题。

鸣谢

我的台式机。

参考

Sui Z, Chen Y. 2002. *The research on the automatic term extraction in the domain of information science and technology*. Proc of the 5th East Asia Forum of the Terminology.

Jun-ichi Aoe, Katsushi Morimoto, and Takashi Sato. 1992. *An efficient implementation of trie structures.* Softw. Pract. Exper. 22, 9 (Sept. 1992), 695–721. DOI:https://doi.org/10.1002/spe.4380220902

Nanyuan L, 1987. *CDWS The Mordern Printed Chinese Distinguishing Word System*. J Chin Inf Proc. 1, 2(1987), 44-52

ICT, 2010. [*ICTCLAS Chinese Distinguishing Word System.*](http://ictclas.nlpir.org/)