分词词典与算法的改进

XXXX

Abstract

中文分词指将中文句子分割成独立的单词，是中文自然语言处理的基本任务之一。传统的机械分词系统简单易用，但是严重依赖词典性能与匹配规则。我们使用前缀树对词典进行了改进，大大加速了查找速度，并基于统计的最大概率分词方法，利用动态规划思想得到概率最优的切分。我们在三个月的人民日报语料上对最大概率分词进行了测试，相较于简单词典机械分词，得到了更快的推理速度与更好的分词精度。

1 绪论

分词是将连续的字序列按照一定的规范重新组合成语义独立词序列的过程。中文虽然字、句、段之间有分割符号，但中文词语之间没有空格进行分割，而自然语言处理中通常以词为最小的处理单位，因此需要对中文进行分词（Word Segmentation）处理。汉语分词是中文信息处理领域的基础课题，也是智能化中文信息处理的关键 (骆正清 and 陈增武, 1997)

中文分词面临两个主要问题，一是切分歧义问题，二是未登录词问题 (?)。切分歧义问题是指同一个句子可能有多个切分结果，例如“乒乓球拍卖完了”，可以切分成“乒乓球”“拍卖”“完了”，或者“乒乓球拍”“卖完”“了”。未登录词问题指的是，因为词典不可能收录所有单词，如人名，地名，机构名等，所以会出应该被切分出的词语却不在词典中的现象。

2 研究现状

目前已知的分词算法有三类：基于词典匹配的分词方法，基于统计的分词方法，基于神经网络的分词方法。

基于词典匹配分词又叫做机械分词，是按照指定规则根据词典对句子中的字符进行匹配，常见的有正向最大匹配、逆向最大匹配方法，这类方法最早出现，也最简单，但是不能很好地处理切分歧义与未登录词的问题。基于统计的分词方法基于大量的已分词文本，利用统计模型学习语言切分的规律，从而实现对未知文本的切分，典型的有隐马尔可夫模型 (Eddy, 1996)，条件随机场模型 (Peng et al., 2004)，基于统计的分词方法在分词精度上有了很大的提升，但是仍然存在跨领域性能下降的问题。由于领域内的词典获取比领域语料获取简单得多，所以基于统计的方法在跨领域能力方面弱于于基于机械分词的方法 (张梅山 et al., 2012)。还有部分工作探索在词典与统计结合的方法 (蒋建洪 et al., 2012)，在跨领域时加入领域词典以提升综合性能。近期，随着语料规模的扩大与计算资源的提升，深度学习方法开始成为主流方法。基于神经网络的分词方法利用深度神经网络学习文本的特征，更好地对文本进行分割，典型的技术有 RNN (Chen et al., 2015)，LSTM (Yao and Huang, 2016)。

本实验聚焦于前向最大匹配，后向最大匹配，最大概率分词的实现与改进。

3 方法与模型

3.1 前向最大匹配与后向最大匹配分词

正向匹配算法从字符串左端开始，按照词典中词的最大长度分割，将分割的子字符串与词典匹配，若匹配成功则使用余下的字符串继续匹配。否则将子字符串从末尾去除一个字符，再进行匹配，直到满足条件为止。

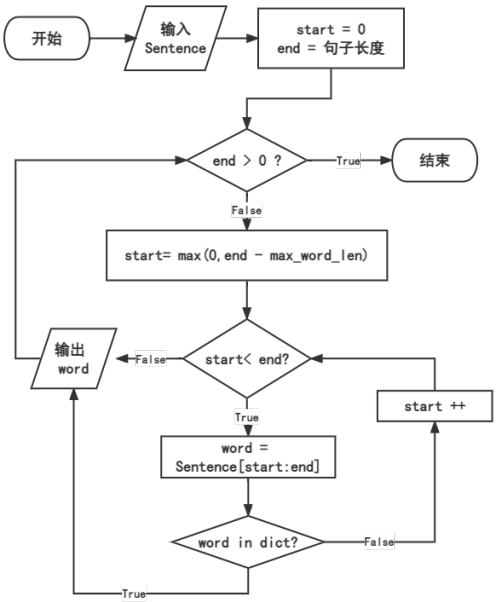

Figure 1: 前向最大匹配分词算法流程图

前向最大匹配分词的最大缺点在于倾向于切分出较长的词，导致错误的切分结果。例如“研究生命的起源”，由于“研究生”是词典中的词，所以分词结果为“研究生 命 的 起源”。

后向匹配算法与正向匹配很类似，算法唯一的不同就是，反向匹配算法是从字符串右端开始的。由于中文的后缀区分度更高的特性，反向最大匹配算法在中文分词任务上一般效果优于前向最大匹配分词算法。

假设使用基于顺序查找的词典，词典长度为 N，则每一次查询字典的复杂度为 $O(N)$。假设词典最大单词长度为 $MaxL$，平均句子长度为 S，前向匹配算法的平均词典查询次数为 $\frac{MaxL}{2} * S$。所以前向算法的时间复杂度为 $O(N * MaxL * S)$。

3.2 基于前缀树改进的词典

若直接使用顺序查找的词典，分词所需时间很长。将词典优化为前缀树格式，能提升分词系统的整体性能。

前缀树又称字典树，是一种树形结构，可以用来统计与排序大量的字符串，其核心思想是空间换时间，利用字符串的公共前缀来减少查询时间。前缀树的基本单位为节点，一个节点表示一个字，其父亲节点唯一。从根节点到终端节点的路径所组成的字符串就是字典中的一个词，终端节点在构建词典时用一个布尔值标定。下图为 [“广州”, “广播”, “人民路”, “人生”, “人民”, “玩耍”] 所构成的前缀树。

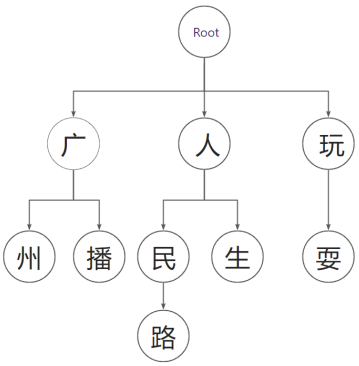

Figure 2: 前缀树

假设词典总单词数为 N，平均单词长度为 W。顺序格式的词典查找的时间复杂度与空间复杂度都很高为 $O(N)$。前缀树格式的词典对此进行了改进，用前缀树对单词进行表示与存储，将查找的时间复杂度降为了 $O(W)$，大幅度提升了查找的速度。

3.3 最大概率分词

基于机械正向最大匹配的分词，是按照最大长度切分词语，可能不是最优切分 (Lin) 。本部分将利用动态规划思想，实现最大概率分词。

最大概率分词是一种基本的统计分词方法，核心思想是利用标注数据统计得到前缀词典后，利用前缀词典对输入句子进行切分得到词图，再通过动态规划计算得到词图上的概率

最大的路径作为切分结果。

设 $C = c_1, c_2...c_n$ 为待切分句子，其中 c_i 为一个字符；$S = w_1, w_2...w_m$ 为分词结果，w_i 代表一个词。最佳切分方法为

$$Seg(C) = argmax_s P(S|C) = argmax_s \frac{P(C|S)P(S)}{P(C)},$$

由于待切分句子是固定的，$P(C), P(C|S)$ 相当于一个常数，所以也可写成

$$Seg(C) = argmax_s P(S)。$$

作一元文法假设，即每个词都是上下文无关的，所以

$$P(S) = P(w_1, w_2...w_m) = \prod_{i=0}^{m} P(w_i)。$$

其中，$P(w_i)$ 为单词在训练语料中出现的频率。通常使用取对数防止连乘带来的概率下溢问题，即

$$Seg(C) = argmax_s log(P(S)) = argmax_s \sum_i^m logP(w_i)。$$

倘若直接暴力遍历所有的切分方式，再找概率最大的切分，其复杂度是 $O(2^n)$ 这种方法是不可行的。为了找到最优切分方法，需要引入前缀词典并构造词图。再使用动态规划求解。

3.3.1 前缀词典

通过前缀词典，可以更方便地构建句子的词图。前缀词典与前缀树词典很相似，除了保存了普通词的词频，也保存了每个词的所有前缀的词频，不同之处在于，前缀词典直接通过哈希保存，且若前缀没有出现过，则将词频赋值为 0。前缀词典格式为 {前缀词: 前缀词频}，通过 Python 内置的 Dict 数据结构保存。

3.3.2 词图

词图是一种有向无环图（DAG），使用内部路径来表示一个句子中所有单词切分。

假设前缀词典中非零频词语为 [“去”, “北京”, “北京大学”, “大学”, “玩”]，那么 “去北京大学玩” 的词图可以表示为图3。

词图的构建可以视为一个有向无环图的构建，其算法如算法1：

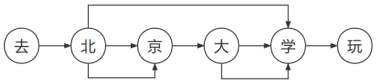

Figure 3: “去北京大学玩” 的词图

Algorithm 1 DAG Construction

Input: 待分词句子 S = $\{w_1, w_2...w_N\}$, 词典 D

$DAG \Leftarrow \{\}$
for $k = 0, 1, ..., N$ do
  $i \Leftarrow k$
  $PossibleIndex \Leftarrow []$
  $Fragment \Leftarrow S[k]$
  while $i < N$ & $Fragment \in D$ do
    if $D.get(Fragment) > 0$ then
      $PossibleIndex \Leftarrow PossibleIndex \cup \{i\}$
    end if
    $i \Leftarrow i + 1$
    $Fragment \Leftarrow Sentence[k : i + 1]$
  end while
  if $Possible_Index \neq None$ then
    $Possible_Index \Leftarrow k$
  end if
  $DAG \Leftarrow DAG \cup \{k, PossibleIndex\}$
end for
return DAG

DAG（词图）中，每一个节点代表一个字符，每一条边代表可能出现的切分单元。DAG 词图利用词典进一步缩小切分搜索的范围。

3.3.3 动态规划

现在，只需找到 DAG 词图中概率最大的路径，即可得到切分结果。

在一元文法的假设下，对于某条路径对应的切分结果 $S = w_1, w_2...w_m$，其概率定义为 $\prod_{i=0}^{m} P(w_i)$。此时问题转化为了 DAG 图的最大加权路径问题，可以通过动态规划解决，其动态规划方程描述为：对于整个句子的最优路径 $S_{best} = w_1, w_2....w_m$ 与末端节点 w_m，其可能存在所有前驱节点 w_i, w_j, w_k 的最优路径分别

Algorithm 2 Max Prob Segmentation

Input: 待分词句子 S = $\{w_1, w_2...w_N\}$, 前缀词典 Dict, 词图 DAG

$Route[N] \Leftarrow (0, 0)$ // Route[i][0],Route[i][1] 分别代表 S 第 i 个位置最优分词结果位置与概率

for $i = N-1, N-2, ..., 0$ do

    $\mathcal{N}_i$ 为第 i 个字符在 DAG 中的所有后继节点

    for x in $\mathcal{N}_i$ do

        使用 $S[i : x+1]$ 得到得到可能的字词切分 $Fragment$

        使用 $Dict.get(Fragment)$ 得到词语的概率 $charFreq$

        防止下溢取对数 $log(charFreq)$ 得到 $logCharFreq$

        更新当前路径的概率值 $AccFreq \Leftarrow AccFreq + Route[x+1][0]$

        if $AccFreq > Route[i][0]$ then

            $Route[idx] \Leftarrow (AccFreq, x)$

        end if

    end for

end for

return $Route$

设为 $S_{best_i}, S_{best_j}, S_{best_k}$, 动态规划方程为:

$$S_{best_m} = max(S_{best_i}, S_{best_j}, S_{best_k}) + P(w_m)$$

为了解决小概率连乘的下溢问题，对概率取对数相加，见算法2。

3.4 前处理与后处理

前处理与后处理是基于手工规则在模型输入前后进行额外处理的方法，在数据分布特点较明显时，可以缓解未登录词的问题，有效地提升分词的精度。

针对给定数据集每一行带序号的特点，我们使用正则表达式对序号进行了过滤，将其替换为一个特定的字符作为模型输入，待分词结束后再将其替换为原有的序号。当数据不含序号时，不影响分词程序的运行，当数据含未登录序号时，能够大幅度提升分词效果。此外，还将序号的过滤扩展为其他未登录词的过滤，如利用人名词典过滤人名。

4 实验与分析

4.1 评价指标

本实验使用准确度（Precision）、召回率（Recall）、F 值（F Score）作为分词评价指标。计算公式如下:

$$P = \frac{A}{B} * 100\%$$

$$R = \frac{A}{C} * 100\%$$

$$F\ Score = \frac{2P*R}{P+R}$$

其中 A 代表切分结果中正确的分词数，B 代表切分结果中所有的分词数，C 代表标准答案中所有的分词数。

对于词典性能的评价，本文使用一个月数据集上分词所需时间 Timecost(s) 作为评价指标。

4.2 实验结果与分析

我们统计得到，单月数据集（199801segpos.txt）中平均句子长度为 96，词典中最大单词长度为 7.8，最大单词长度为 27。前向最大匹配分词平均每句的词典

		训练集效果			测试集效果		
Re	split	Precision	Recall	F Score	Precision	Recall	F Score
No	10%	98.82%	97.89%	98.36%	72.34%	94.81%	82.02%
	20%	98.86%	97.96%	98.41%	71.98%	94.70%	81.79%
	30%	98.86%	97.96%	98.41%	71.99%	94.64%	81.76%
Yes	10%	98.88%	97.88%	98.35%	95.52%	96.41%	95.96%
	20%	98.87%	97.98%	98.43%	95.53%	96.42%	95.87%
	30%	98.86%	97.97%	98.42%	95.10%	96.32%	95.71%

Table 1: 最大概率分词在完整数据集上的简单交叉验证，测试集采样率分别为 10%,20%,30%，*Re* 指前处理与后处理

查询次数为 1212 次，符合3.1的理论分析结果，同时也证明了词典是性能优化的关键。

	顺序字典	前缀树字典
FMM	72031.57	944.81
BMM	>58000	875.54

Table 2: 在一个月数据集下，前向最大匹配 FMM 与后向最大匹配 BMM 在不同字典实现对应的分词时间 (s)

我们将词典查找方式由顺序查找改进为前缀树查找，并在单月数据集（199801segpos.txt）下，利用前向最大匹配分词 (FMM) 与后向最大匹配分词 (BMM) 对词典性能进行了评估，结果如表2。可以看出，前缀树查找的方式相较于顺序匹配的方法有大于 70 倍的速度提升。

我们实现了基于 DAG 的最大概率分词模型，并在单月数据集下进行了封闭测试 (训练与测试文件均为 199801segpos.txt)，与机械匹配分词方法 (FMM 与 BMM) 进行了对比，结果见表3。由于封闭测试，三者的的准确率、召回率、F 值都很高。BMM 比 FMM 综合性能更好，这是因为在中文逆向匹配时遇到的歧义更少。DAG 的效果最好的准确率、召回率、F 值均优于 FMM、BMM，可见统计模型方法效果优于机械分词。

我们在完整的三个月数据集上对最大概率分词模型的泛化性进一步进行了评估，基于随机采样分别进行了简单交叉验证，结果如表1。可见，模型的泛化性能并不是很好，在测试集上模型的准确率下降了 27%，召回率下降了 3%。此外，训练集划分比例的扩大，会对测试集的性能带来轻微提升。

封闭测试	一个月数据		
	Precision	Recall	F
FMM	97.71%	97.06%	97.38%
BMM	97.88%	97.24%	97.56%
DAG	99.01%	98.43%	98.71%

Table 3: 在一个月数据集封闭测试下（训练集与测试集相同），前向最大匹配（FMM），后向最大匹配（BMM）与最大概率分词（DAG）的效果

我们探究了模型泛化性低的原因，对错误分词测试用例进行了分析，发现了很多数字、人名的未登录词的分词错误，其中错误占比最大的是句子开头的序号，如 “19980318-06-004-003”，由于词典中未收录，会直接按单个字符进行分割，而不是视为整体。此外第三个月的语料中出现了大量的人名，造成了未登录词过多的问题，对分词性能也有一定的影响。

加入前处理与后处理后，我们再次进行了实验，模型在测试集的精度上有了明显的提升。

5 结论

本文探究了词典与分词算法的改进，在前向最大匹配分词与后向最大匹配分词的基础上，使用前缀树加快了词典的查找速度，并实现了最大概率分词算法，得到了相比机械分词更好的效果。同时，本文对最大概率分词模型进行了交叉验证，探究了所实现模型泛化能力差的原因并进行了改进。

References

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, and Xuan-Jing Huang. 2015. Gated recursive neural network for chinese word segmentation. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1744–1753.

Sean R Eddy. 1996. Hidden markov models. Current opinion in structural biology, 6(3):361–365.

ShiZhong Lin. Blog: Chinese segmentation. [EB/OL]. http://www.shizhuolin.com/2018/01/21/1860.html/ Accessed January 21, 2018.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics, pages 562–568.

Yushi Yao and Zheng Huang. 2016. Bi-directional lstm recurrent neural network for chinese word segmentation. In International conference on neural information processing, pages 345–353. Springer.

张梅山, 邓知龙, 车万翔, and 刘挺. 2012. 统计与词典相结合的领域自适应中文分词. 中文信息学报, 26(2):8–13.

蒋建洪, 赵嵩正, and 罗玫. 2012. 词典与统计方法结合的中文分词模型研究及应用. 计算机工程与设计, 33(1):387–391.

骆正清 and 陈增武. 1997. 汉语自动分词研究综述. 浙江大学学报: 自然科学版, 31(3):306–312.