汉语分词系统

120L021902 敬刘畅(无分组)
哈尔滨工业大学

摘要

本次实验的目标是构建一个汉语分词系统，并全面掌握汉语分词的若干相关关键技术。基于上述目标，本次实验建立了基于机械匹配分词的正反向最大分词匹配方法分词系统、基于统计语言模型的 N 元文法分词系统以及基于 HMM 的未登录词识别系统。最终完成了二元文法与 HMM 相结合的分词系统，并在测试集上获得了 0.95 的分词精确率。

1 绪论

分词是自然语言处理的基础任务之一，它指的是将一个句子序列切分成一个一个单独的词。如果将语言的单位从小到大分为字符级、词级、和句子级，则分词任务就是要从句子级的数据中切分出符合语义的句子级信息。而对于像英语这样的语言，其句子由独立的被空格分隔的词构成，而词再由独立字符构成，因此不存在分词问题。但是对于中文类似的语言，其句子是直接由连续的字符组成的，并不能显式地反映出词级信息，因而就诞生出了分词问题。

目前，主流的分词方法可以分为三类：1.基于匹配的方法；2.基于统计的方法；3.基于深度学习的方法。本文中实现了其中前两种分词方法，包括：

1. 机械匹配方法分词的词典构建
2. 基于匹配的 FMM 和 BMM 分词方法实现与优化
3. 基于 MM 统计模型的二元文法分词实现
4. 基于 HMM 的未登录词识别
5. 基于二阶 HMM 的字符级分词方法实现 document

最终，效果优化后的分词系统在开放测试中获得了 97%以上的精确率。

2 相关工作

较早的有效分词模型是上世纪 80 年代提出的一系列基于词表机械匹配的方法，例如：正向最大匹配法、逆向最大匹配法、双向最大匹配法等。机械匹配的方法使用简单的方法却获得了较好的效果，因而针对其的进一步研究与改进的相关工作[1, 2]直到近年也一直在进行中，并取得了不错的效果。

在基于统计的方法出现了后，分词的效果获得了进一步的提升，其中的典型模型包括 HMM 和 CRF 模型。同时也出现了一批效果优秀的分词系统，其中以 jieba 与 SnowNlp 为代表。Jieba 分词主要依赖于 N 元文法与 HMM 未登录词处理的方式，而 SnowNlp 则是依赖于三阶 HMM 对待分词句子进行序列标注后再进行分词。

2017 年以来，随着深度学习模型的兴起，统计方法和深度学习模型的融合成为了新趋势之一，其中的代表工作是 CRF+Bi-LSTM 模型[3]。2018 年预训练模型 BERT 出现后，更多的工作出现在了基于预训练模型的分词方法上。

3 实现过程

3.1 词典的构建

词典是许多分词方法的基础，词典的构建好坏将一定程度上影响分词模型尤其是机械匹配模型的分词速度、分词效果和泛化能力。因此，在构建词典时应当遵循以下的一些规则和标准：

- 选择性：选择性主要从词长度和词泛化能力两个方面考量。对于词长度过长的词，例如网站域名、特称词、英文、少数民族人名等等，这一类词通常出现概率较小，如果将其加入词典中，对于机械匹配模型如 FMM，一方面增加了最大匹配的长度导致分词时间增加，一方面又难以获得更好的分词性能。对于词泛化能力差的词，例如“1998 年”、“50 吨”、“19980325”等等，这一类词的语义并不具备泛化能力，将其加入词典中，将会导致过拟合的问题。因此，构建词典时，将超过 10 的长词、纯符号词、量词等去除。

- 结构性：针对不同的模型需求，词典应当具有不同的结构。例如，对于机械匹配分词，词典中元素是互不相同词，同时词典还应当包含最大匹配长度的大小信

息，使得分词模型读取词典后就能够自动获取模型参数。而对于基于统计的 N 元模型分词，词典中则应当包含互不相同的 N 元对以及其出现频率。对基于序列标注（字成词）的统计模型，词典中除了词、词频还应当包括该词的词性或者标注信息。

3.2 正反向最大匹配分词

最大匹配分词认为长度较大的词有更多的语义信息，每次选择可能切分中长度最大的词，选择的依据来自于词典的匹配。

正向最大匹配算法 FMM：FMM 分词过程中每次匹配的最大词长度来自其词典中的最长词的长度。从句首开始，先匹配一段与最大词长度相等的字符串，若其在字典中则将该字符串作为一次切分结果，否则去掉该字符串的最后一个字符，再到词典中匹配，直至匹配成功或者字符串中仅剩下一个字符单字成词。算法流程图如图 1 所示。

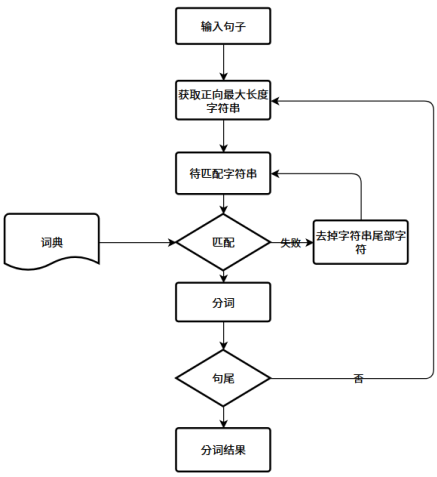

图 1.FMM 算法流程图

反向最大匹配分词 BMM：BMM 的分词过程与 FMM 类似，不同之处在于 FMM 从词首开始匹配，每次匹配失败删除字符串的最后一个字符，而 BMM 从词尾开始匹配，每次匹配失败删除字符串的第一个字符。其算法流程图如图 2

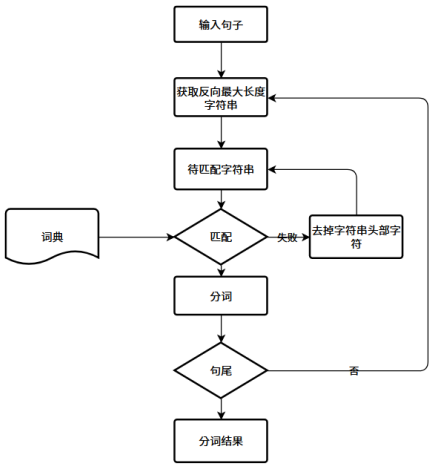

图 2.BMM 算法流程图

实验以最少代码量的方法分别实现了 FMM 和 BMM。即：将字典读入 list 数据类型中，每次匹配时通过线性循环的方式遍历列表来查找是否匹配成功。

实验发现，正反向最大匹配分词算法虽然实现简单，但是其精确率在开放测试时也能达到 95%以上。但是其匹配词典的过程如果没有优化后的查找算法，匹配时间将会非常长，在实验测试集上完成分词实验需要约 6 小时的时间，因此需要采取一定的优化措施。

3.3 正反向最大匹配分词性能评价

实验采用了滑动窗口法为 FMM 和 BMM 的分词结果与标准结果进行了比较，并计算了其精确率 P、召回率 R 和 F1 值，计算公式如下：

$$P = \frac{TP}{TP + FP} = \frac{\text{分词结果中正确分词数}}{\text{分词结果中所有分词数}}$$

$$R = \frac{TP}{TP + FN} = \frac{\textit{分词结果中正确分词数}}{\textit{答案中所有分词数}}$$

$$F1 = \frac{2PR}{P + R}$$

从计算结果上看，BMM 略微优于 FMM。这是中文的重点词偏后导致的：中文句子的重点词通常在句子后部，例如：“扣篮球”重点在“篮球”，“执法部门”重点在“部门”，而 FMM 因为是正向匹配将更容易抓取错重点。例如在图 3 的分词结果中可以看到，对于“无法用语言”，FMM 将其切分成“无法”“用语”“言”，BMM 将其切分成“无法”“用”“语言”。

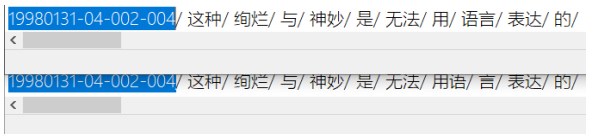

图 3.FMM 与 BMM 对比

从具体分词结果上看，FMM 和 BMM 对分词歧义尤其是交集型歧义处理效果一般。FMM 常常出现将更符合语义的下文切分开的情况，而 BMM 则常常出现将更符合语义的上文切分开的情况。这是 FMM 和 BMM 的匹配算法导致的：匹配算法使得 FMM 和 BMM 只要匹配到了最长的串就立会将其作为一个切分，而无法考虑到也许下一个匹配窗口的切分结果可能优于当前切分的情况。

.

3.4 FMM 和 BMM 的速度优化

FMM 和 BMM 的速度优化重点在于优化匹配词典时的查找算法。实验采用了基于 trie 树与 hash 表相结合的查找算法。

Trie 树也称为字典树，它将一个字符串转化为树上的一条路径，其每个节点包括当前节点代表的字符、是否是终结节点和子节点信息，从根节点到某终结节点的一条路径就代表了字典中的一个词。为了进一步加快 Trie 树的查找速度，利用除余法哈希函数和线性探测处理冲突的方法为每个节点的子节点信息建立了 hash 表。最终 trie 树的结构和 trie 树查找算法分别如图 4、图 5 所示。

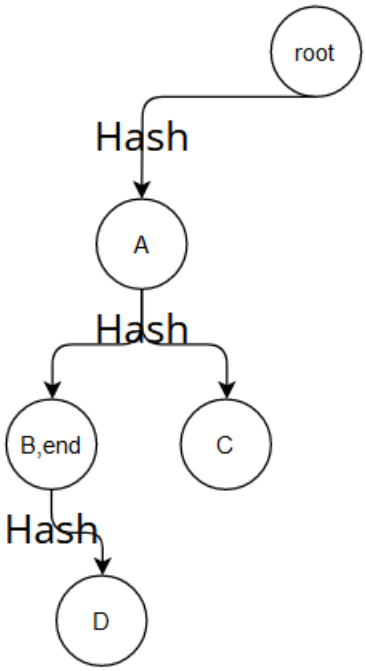

图 4.结合 Hash 的 trie 树结构存储 AB 词汇

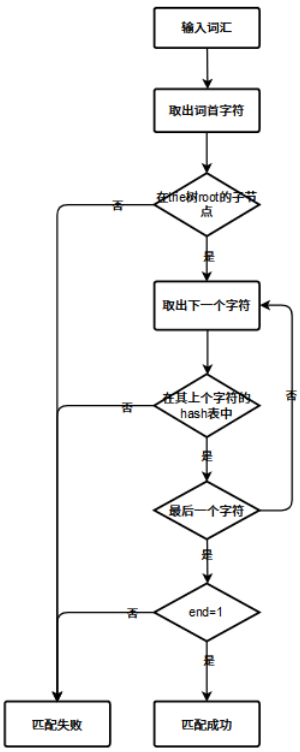

图 5.trie 树查找算法

优化结果表明，采用 trie 树结合 hash 的方式 FMM 和 BMM 分词都能在 2 分半之内完成测试集分词任务，速度提升了约 370 倍。

3.5 基于统计语言模型的中文分词

实验中实现了基于统计的二元文法语言模型分词任务，并在性能上相较 FMM 和 BMM 获得了进一步提高。

二元文法语言模型：统计语言模型的任务可以描述为：给定一个 k 个词组成的句子，计算其出现的概率，数学描述如下：

$$P(w_1 w_2 \dots \dots w_k) = ?$$

结合条件概率公式，可以改写为：

$$P(w_1 w_2 \dots w_k) = P(w_1)P(w_2|w_1) \dots P(w_k|w_1 w_2 \dots w_{k-1})$$

显然，此时参数空间极大，条件概率$P(w_k|w_1 w_2 \dots w_{k-1})$难以估算。因此我们引入 2 元语言模型。2 元语言模型认为某个词出现的概率仅仅依赖于它的上一个词，于是我们有：

$$P(w_1 w_2 \dots w_k) = P(w_1)\prod P(w_n|w_{n-1})$$

显然，通过最大似然估计可以很容易地根据二元对(w_{n-1}, w_n)的出现次数估计$P(w_n|w_{n-1})$。

二元文法分词系统：二元文法分词系统的实现依靠以下三个部分：一元、二元前缀词典构建、生成全切分有向无环图(DAG)、动态规划求最大概率路径。其分词流程图如图 6 所示

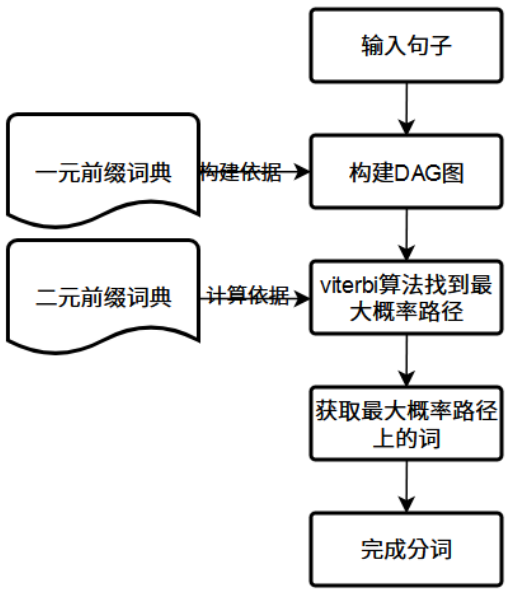

图 6.二元文法分词

一元、二元前缀词典构建：二元前缀词典就是二元对：平滑后二元对出现频次的形式，无需赘述。而一元前缀词典在获取了一元：一元出现频次后还需对每个一元词的前缀词进行处理，如果已经存在于前缀词典中，则不处理，

如果该前缀词不在前缀词典中，则将其词频置为 0，便于后续构建有向无环图。构建一元前缀词典算法如下：

算法：一元前缀词典构建

输入：训练语料 data

输出：一元前缀词典 lfreq 和总词数 ltotal

```
for word,count in data do:
    lfreq[word]=count
    ltotal+=count
    for prefix of word:
        if prefix not in lfreq:
            lfreq[prefix]=0
        end if
    end for
end for
```

生成全切分有向图：全切分有向图结构为{k : [k , j , ..] , m : [m , p , q] , ...}的字典结构，其中 k : [k , j , ..]表示句子中，从 k 位置开始可以切分的词是 k~k、k~j…。构建全切分有向图只需从句首开始遍历每一个位置，在此位置取出所有能和它组成一个在前缀词典中的词典后续，并将其记录为一条路径，直至遍历到句尾，就生成了句子的全切分有向图，其中每一个从句首到句尾的路径就是一种切分方式。

动态规划求最大概率路径：实验中使用维特比算法求 DAG 中的最大概率路径，维特比算法如下

算法：维特比算法

输入：观测序列 T，状态空间 N

输出：最大概率路径

```
v [N,T]
for each state s from 1 to N do
    v [s,1] ← πs*bs(o1)
    p [s,1] ← 0
for each time step t from 2 to T do
    for each state s from 1 to N do
    v [s,t]←max v[s',t]as'bs(ot)
    p [s,t]←argmax v[s',t]as'bs(ot)
prob ←max v [s,T]
p←argmax v [s,T]
bestpath ←随 p 逆序的状态序列
return bestpath,bestpathprob
```

3.6 性能优化

基于 HMM 的未登录词识别: HMM 的基本任务是给定观察序列$O=\{o_0,o_1,\dots o_n\}$，找到能够最好地解释O的状态序列 S={ $o_i,o_j,\dots o_k$ }。在本任务中，状态集有 Q={B,M,E,S}，分别代表词首、词中、词尾和单自成词，HMM 的任务是根据输入的句子给句子里的每一个字打上状态集 Q 中的标签。HMM 标注算法如下：

算法：HMM

输入：字符序列 O 模型参数λ=(A,B,π)状态集 Q

输出：状态序列

```
for i ←1,2,3...N do
    δ1(i) = πi bi(O1)  φ1(i) = 0
end for
for i ←1,2.....N，t←2,3.....T do
δt(i) = max_j δt−1(i) aji bi(Ot)
φt(i) = argmax_j δt−1(i) aji bi(Ot)
end for
P = max_i δT(i)  qT = argmax_i δT(i)
qt = φt+1(qt+1)得状态序列Q
return Q
```

其中 HMM 的模型参数λ=(A,B,π)可以由最大似然估计方法根据数据集训练得到。

HMM 为句子完成标注之后，按照 B,M,E,S 的含义即可完成对句子的分词，这就是 HMM 的字构词分词方法，其优点是不会出现遇到未登录的情况，训练完成后即使是没有词典也可以正确分词。

但是单纯的 HMM 字构词方法精确率较低，因此实验中将其用作二元文法分词的未登录词模块。因为二元文法分词对于未登录词会将其切分成一个一个的单字成词，因此，对于二元文法中的连续的单字成词部分，将这些单字合并为字符串后使用 HMM 字构词的方法重新切分后再重新并入二元文法分词结果中完成未登录词识别。

基于二阶 HMM 的分词系统:HMM 使用了类似二元文法的假设，每次只能考虑当前词的前一次，对于上下文信息很难涵盖。而二阶 HMM 类似于三元文法，每次能考虑前两个词的信息，对上下文信息的获取能力强于一阶 HMM。因此可以使用二阶 HMM 来优化分词性能。

由于二阶 HMM 依赖于前两个状态，因此，其标注模型可以表示为：

$$arg\max_t [\prod P(t_i|t_{i-1},t_{i-2})P(c_i|t_i)]P(t_{n+1}|t_n)$$

其中，t 是 tag 也就是字符的 BMES 标签。从上式可见，二阶 HMM 与一阶 HMM 唯一的区别就是计算当前字符的最大概率标签时需要考虑前两个标签。计算上与一阶 HMM 唯一不同的就是计算条件概率：$P(t_i|t_{i-1},t_{i-2})$。

其计算方法如下：

$$P(t_3|t_2,t_1)$$
$$=\lambda_1 P(t_3)+\lambda_2 P(t_3|t_2)+\lambda_3 P(t_3|t_2,t_1)$$

其中，$P(t_3)$，$P(t_3|t_2)$，$P(t_3|t_2,t_1)$可以通过极大似然估计得到，而λ_1，λ_2，λ_3是二阶 HMM 相对于一阶 HMM 需要多训练的参数。

其训练方法如下：

算法：λ参数训练

输入：一元词典、二元词典、三元词典

输出：λ参数

```
λ1=λ2=λ3=0
for trigram in trigramdict
  uni=P(t3)bi=P(t3|t2)tri=P(w3|w1,w2)
    find maxinum in (uni,bi,tri) do
      if maxinum=uni do λ1+=P(t1,t2,t3)
      if maxinum=bi do λ2+=P(t1,t2,t3)
      if maxinum=tri do λ3+=P(t1,t2,t3)
end for
```

4 实验结果

4.1 FMM 与 BMM 性能比较

指标	FMM	BMM
P	0.958	0.960
R	0.943	0.945
F1	0.950	0.952

表 1 FMM 与 BMM 性能比较

测试发现 BMM 各项性能均略优于 FMM，这是汉语的重点词偏后导致的。

4.2 FMM 与 BMM 优化后分词用时比较

查找方法	FMM 耗时(s)	BMM 耗时(s)
List 循环	59010	58130
Trie 树	192.62	158.47

表 2 优化前后分词耗时比较

测试发现，基于 trie 树+hash 函数的优化方法能够将分词速度提高约 370 倍。

4.3 各分词方法精确率比较

	Fmm	bmm	Bigram	Oov	biHMM
P	0.932	0.936	0.945	0.953	0.974

表 3 各方法分词性能比较

可见，统计语言模型普遍性能优于机械最大匹配模型，同时，实验中采用的各种优化手段包括 HMM 处理 OOV 和二阶 HMM 方法对于提高分词系统性能是有效的

References

[1]. Qu, H.Y. and W. Zhao, A Revised BMM and RMM Algorithm of Chinese Automatic Words Segmentation, in MANUFACTURING SYSTEMS AND INDUSTRY APPLICATIONS, Y.W. Wu, Y.W. Wu^Editors. 2011: International Conference on Materials Engineering for Advanced Technologies (ICMEAT2011). p. 199-204.

[2]. Tang, J., Q. Wu and Y.H. Li, An Optimization Algorithm of Chinese Word Segmentation Based on Dictionary, in 2015 INTERNATIONAL CONFERENCE ON NETWORK AND INFORMATION SYSTEMS FOR COMPUTERS (ICNISC), Z. Qian and H. Wang, Z. Qian and H. Wang^Editors. 2015: International Conference on Network and Information Systems for Computers (ICNISC). p. 259-262.

[3]. Ma, X.Z. and E. Hovy, End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF, in PROCEEDINGS OF THE 54TH ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, VOL 1, K. Erk and N.A. Smith, K. Erk and N.A. Smith^Editors. 2016: 54th Annual Meeting of the Association-for-Computational-Linguistics (ACL). p. 1064-1074.