汉语分词系统实验报告

Khadorstorm
khador@foxmail.com

摘要

分词是汉语自然语言处理中的基本任务, 分词结果的质量很大程度决定了后续任务的质量。本次试验中实现了从语料库构建分词词典, 正反向最大匹配分词算法, 最大词频分词算法。在此基础上，又编写了可以处理未登录词的基于字的 HMM 模型和二元版本的基于字的生成式模型 (Character-Based Genarative Model)。最后，设计了一个 CBGM 与 BMM 结合并做后处理的分词模型，并对以上模型的实现思路和性能做了一些评价分析。

1 绪论

词是最小的能够独立运用的语言基本单位，而与西方语言不同，中文在书写时没有分隔符指明词之间的边界。因此，分词任务成了汉语自然语言处理的首要任务。自动分词是正确的中文信息处理的基础。(宗成庆, 2008)

分词任务主要面临如下问题: 首先, 对‘词’的划分标准就有待商榷；其次，在给定划分标准下仍会存在歧义切分问题和未登录词问题。接下来我们对一些经典解决方案进行介绍和梳理。

2 相关工作

最早期分词方案是基于机械匹配的算法, 按照一定顺序和条件将带分词语句的一部分与预先建立的词典进行字符串匹配。具体的算法有最大匹配法、双向匹配法、最少分词法和最大词频法等。

之后提出的是基于统计语言模型的算法, 包括基于 n 元语法的分词和基于 HMM 的分词方法等。又可以分为基于词的方法和由字构词的方法。

无论是机械匹配还是基于词的方法，模型本身都几乎没有能力处理未登录词问题，需要引入额外的未登录词识别模块。而由字构词的方法将分词问题看作是对句子中每个单字的分类问题，因此能够平衡地看待词表词与未登录词，在有较好性能的同时也简化了分词系统的设计。

通常而言，基于词的方法使用的都是生成式模型，而由字构词方法一般使用的是判别式模型。后者虽在未登录词的召回率上有显著优势，但在词表词的准确率上反而略逊一筹。原因是只使用单字在词中的位置这一信息往往不够充分，而基于词的方法考虑字与字之间的相邻关系, 更符合人们对人类分词方法的认识。为了使有字构词的模型也能利用字临接信息，一种方法是在当前字的上下文中开一个窗口抽取特征，另一种方法是构建基于字的生成式模型, 将（字，词中位置）这一二元组当作计算的基本单位，取得了很好的效果。

本次试验中实现了 FMM，BMM，最大概率分词，基于字的 HMM 和二元版本的基于字的生成式模型, 接下来对这些模型做具体讨论。

3 本文模型和算法

3.1 文本预处理与字典构造

3.1.1 正则预处理

考虑到语料中有时间戳，大写/小写数字, 字母，各种符号混合等会被标注为词，因此在使用语料前先利用正则表达式对这些内容作处

识别内容	替换符号	正则表达式
大写数字	è	((百 | 千 | 万) 分之	第)*(([零-十]| 两 |)+([百-亿]| 点 | 分之 | 比 | 几)*)+
其他符号	§	([A-Z]|[a-z]|[Ａ-Ｚ]|[ａ-ｚ]|[0-9]|[０-９]|-| － |—|%|％|+|.|. |·|:|‖ ／)+

表 1: 正则表达式一览

注意：类似“(百 | 千 | 万) 分之”，“[零-十]”这样的表述其实是不符合正则表达式规则的，这样书写是为了排版方便，请读者领会精神，可具体查看代码

理。具体实现为将每一句话中所有与正则表达式匹配的部分替换成一个未在语料里出现过的特殊符号并返回一个迭代器，在分词结束后将其复原。具体对应关系见表 1。

3.1.2 词典构造

最终的字典包含“词词性词频”三项内容，语料中所有直接可被正则表达式匹配的词不加入字典，但形如“[正则 match]+ 汉字”的词予以保留，如“PC 机”，“IC 卡”等。正则表达式使用的替换符号加入词典，词性设置为 sp，词频设置为一足够大的值。

只有 HMM, BMM 和最大概率分词使用构造的词典，基于字的模型只需先对预料正则处理。对于 HMM 模型，整个 GBK 字库都加入模型，使用 +1 法平滑。

3.2 FMM/BMM

最大匹配分词的原理是，设词典中最长词长度为 i，从当前待处理句子中取出前（或后）i 个字到词典中查找，若找到则切分出一个词，未找到则去掉当前字段的最后（最前）一个字重复上述过程，直到切分成功后重新取下 i 个字循环，直至切分完整个句子。

3.3 最大概率分词

最大概率分词将分词问题抽象成一个在由一个语句所有可能切分构成的有向无环图上搜索最大概率路径的问题。由于需要所有可能的切分，所以需要首先构建前缀词典，即将每个词典词及其频率加入前缀词典，对于该词的所有前缀，若也是一个词典词则不处理，若不是词典词则将其词频置 0 加入前缀词典。

对每个句子 line 查询词典可构建出有向无环图，有向无环图用字典结构存储，其每个键值对为 k:[k, j ,...]，字母均为句子中字的下标，表示 line[k:j+1] 是一个可能的分词

在有向无环图上查找到达第 i 个字的最优路径 R_i，其前驱的结合记为 S 有最优子结构

$$R_i = \max_{j \in S} \{R_j\} \cdot p(w_i) \tag{1}$$

于是可以构建自底向上的动态规划算法求解。

本部分的代码实现参考了课程 PPT 提供的代码。

3.4 基于字的 HMM

隐马尔可夫模型是一个双重随机过程，模型每个时间步按一定概率在隐状态间转换，每个隐状态有各自独立的输出概率分布。我们只知道模型状态转移概率和各个状态下输出的概率分布，而不知道具体的状态序列。

对于基于字的构词方法来说，每个字对应表 3 四种标签中的一种。标签可以看作是隐状态，句子就是观察序列。隐状态的概率转移矩阵和发射概率分布均可通过训练语料进行统计估计，分词问题就可以转化成一个由观察序列求解隐马尔可夫模型的隐状态序列的问题。

这一问题可以由 Viterbi 算法求解，其具体流程是：将最优定义为对于给定模型 μ 和观察序列 O 的情况下使得 $P(Q|O,\mu)$ 最大

$$Q' = argmax_Q P(Q|O,\mu) \tag{2}$$

定义维特比变量 $\delta_t(i)$,

$$\delta_t(i) = \max_{q_1,q_2 \cdots q_{t-1}} P(q_1, q_2 ... q_{t-1} = s_i, O_1 O_2 ... O_t|\mu) \tag{3}$$

标记	含义
S	成词单字
B	词首
M	词中
E	词尾

表 2: 由字构词模型的标签及其含义

表示在时间 t 时，HMM 沿某条路径到达状态 s_i，并输出观察序列 $O_1O_2...O_t$ 的最大概率，其具有最优子结构

$$\delta_{t+1}(i) = \max_j[\delta_t(j) \cdot a_{ij}] \cdot b_i(O_{t+1}) \quad (4)$$

可构建动态规划算法求解。

但是，由于没有利用邻接信息，只使用基础 HMM 构建的分词系统效果并不理想（具体分析见 5.3.1），因此考虑使用考虑字邻接关系的基于字的生成式模型。

3.5 基于字的生成式模型

(Wang et al., 2009) 提出的基于字的生成式模型旨在同时利用基于字的模型对未登录词的识别能力和生成式模型利用字邻接关系的能力，具体而言，其将计算单位变成 [字，标签] 的形式。

3.5.1 模型推导

句中第 i 个字对应的 token 记作 $[c,l]_i$, 在给定字序列 c_1^n 的条件下得到的分词序列 $[c,l]_1^n$ 的概率为

$$P([c,l]_1^n|c_1^n) = \frac{P(c_1^n|[c,l]_1^n) \times P([c,l]_1^n)}{P(c_1^n)} \quad (5)$$

其中 $P(c_1^n|[c,l]_1^n) = 1$, $P(c_1^n)$ 为常数, 因此我们只需考察 $P([c,l]_1^n)$ 即可。本次试验使用二元模型，那么

$$P([c,l]_1^n) = P([c,l]_1) \cdot \prod_{i=2}^{n} P([c,l]_i|[c,l]_{i-1}) \quad (6)$$

其中

$$P([c,l]_i|[c,l]_{i-1}) = \frac{P([c,l]_i,[c,l]_{i-1})}{P([c,l]_{i-1}} \quad (7)$$

可由训练预料统计得出。

3.5.2 动态规划设计分析

基于字的模型虽不需要在全切分有向图上搜索，但本身的搜素空间大小仍在指数级别。对于句子中的每个单字 c，在分词时都有 $[c,S]$, $[c,B]$, $[c,M]$, $[c,E]$ 四种可能性，若句子长度为 n 则搜索空间大小为 4^n, 需要寻找动态规划算法求解。

注意到本模型的搜索过程与 HMM 有相似之处，即每一时间步需要考察的状态数不变，因此类比 HMM 实现中的维特比变量设计动态规划变量 $\psi_t(l)$, 表示使得第 t 个字判定为 l 类的所有分词结果中最大的概率，其中 $l \in L$, $L = \{S,B,M,E\}$. 于是，有递归关系

$$\psi_t(l) = \max_{l' \in L} \psi_{t-1}(l') \cdot P([c_t,l]|[c_{t-1},l']) \quad (8)$$

一句话最大的分词概率 p 为

$$p = \max_{l \in L} \psi_t(l) \quad (9)$$

在此基础上，设计自底向上的动态规划算法，由递归式可以看出每一时间步需要使用上一步的四个 $\psi(l)$。进一步，还需要记录并更新与四个 $\psi(l)$ 对应的分词序列。

最终算法实现，使用一个 [float,list] 列表 best_result 来表示 $\psi(l)$，注意到我们可以在编程时用 0，1，2，3 表示 S,B,M,E，从而可通过 best_result 的下标检索所需的 $\psi(l)$。对句子中的每一个字循环，里面用两层 [0:4] 的循环分别用于计算所有的 $\psi_{t-1}(l') \cdot P([c_t,l]|[c_{t-1},l'])$, 从而找出四个最大值，更新后面标签列表，生成下一个 best_result 供下一个字使用。

3.5.3 数据平滑

在调用二元文法的条件概率 ==0 时，用转为使用当前字一元文法中出现的概率，当一元文法也不命中时（出现未登录字）使用 +1 法作平滑。

3.6 最终的集成模型

最终的集成模型以二元 CBGM 为主体, 当 CBGM 分出的词数/句子字数大于某个阈值时

放弃 CBGM 的结果，用 BMM 重新分词。在分词结果中合并所有“大写年份”，“年”，当单个词长度超过某个阈值时将其拆分成两个词，其中第一个长为阈值。

4 模型训练

训练数据有三个月的人民日报语料，将其均匀划分为 10 份用作十折交叉验证，模 10 同余的行数为一折，最终提交模型使用全部三个月数据训练。

5 实验结果与评价

5.1 词典性能分析

5.1.1 准入的“可枚举”原则与“正则不碰撞”原则

词典不应当收入那些容易识别但不可穷举的词，如数字，字母符号等等。但有一些词汇既能正则又可枚举，这时需要考量正则的实现难度和这些词流入词典的影响。

正则表达式与预料中的特殊序列不是一一对应的，有写内容可能能够被多个正则表达式所识别，如“10 点半”可以用一个识别时间的正则处理，但识别数字序列的正则表达式同样会识别到其中的“10”；对于词“ＨＭＣ５０２２ＸＹＣ”应当被分成一个词，这时就不应让单独识别数字/字母的正则表达式先对其操作。这时就需要我们设计正则表达式的内容和检测顺序以避免潜在的麻烦，以避免影响性能。而另一方面，将时间类不做正则处理在语料较少时又很难将所有情况覆盖全面，也会使得分词性能下降。

由于正则项的替换和复原一旦出错，输入输出的文字序列将有差别，会极大破坏分词性能。最终的处理方案是：正则表达式在设计时遵循不碰撞原则，即正则表达式的使用顺序不会影响最终识别结果。且只对不可枚举的词使用正则，除了年份以外的中间词都保留在字典中，其余的词由于大都形如“【正则】汉字”，可以交由后处理解决。

5.1.2 格式与顺序

最初本着全面的原则，制作了【词 词性 词频】格式的词典，但回头审视发现内容多的方案不一定是最好的方案。除非使用基于词的 HMM 模型，否则并不需要使用词性信息。然而在其他模型中使用一个有词性的词典就需要对同词不同词性的情况做合并，加大了编程工作量。

对于汉字来说，编码顺序并非易读顺序。文件中使用拼音字母序才更有利于手动查找或修改词典，在调用词典时再使用排序函数即可。

5.2 FMM 与 BMM

5.2.1 实现收获

首先是在对内置数据结构的限制下真切体会到了查找算法/索引结构在面对大数据量时的重要性。只使用线性查找，分好一个月的语料需要 5-7 个小时。而只是简单优化成二分查找就可以在一分钟内运行完毕。而课程建议中提供的使用了字典结构的 trie 树只需要 20 秒左右。可见复杂度 $O(1)$ 的哈希优势明显，而设计哈希是一个之前不曾考虑又值得考虑的问题。

其次是最大词长度设置对代码运行时间的影响。代码的循环次数（一定程度上讲也是运行时间）与最大词长度的设置成正比。而词典中的词长度普遍不超过四，可能存在由于词典中有某个极长的人名使得最大词长度变得很大，进而影响算法速度。可以考虑手动设置最大词长为 5-8 左右，以性能的轻微损失换取速度。

最后是代码实现的可移植性，在进行到最终的模型集成时发现将模型写成一个类更有利于提供不同的分词函数（对文件/对单句）用于集成。为此又添加了一个用于模型集成的类 PortableBMM. 同时各个模型也应对外打包成一个对类透明的单独的接受相同参数的函数，方便用户使用。

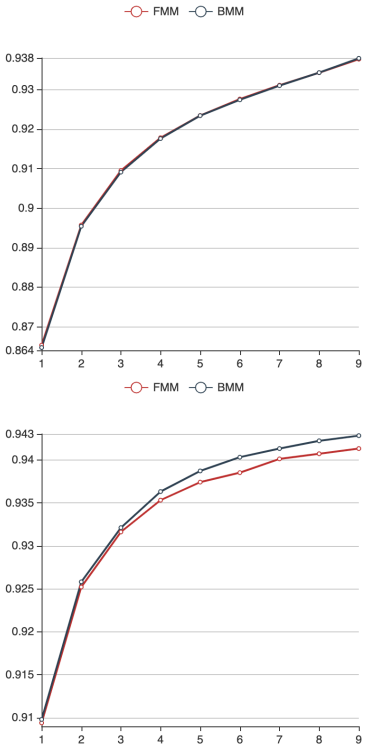

图 1: FMM/BMM 性能比较，上图为只使用 1 月语料，下图为使用全部语料

5.2.2 性能比较与差异

采用累加数据的方法比对 FMM 和 BMM 的性能。即对于已有的十折语料，依次使用一折，两折，..., 九折的数据来测试切分第十折的性能。实验结果见图 1.

从图中可以分析得出，FMM 与 BMM 的性能曲线整体上一致，当训练语料较少时两者性能几乎没有差异，随着训练语料增加，BMM 的性能要稍好于 FMM。

对于 1 月的数据，在 7 折之前 FMM 性能以极小优势好于 BMM，7 折之后 BMM 的表现开始反超。使用全部数据来估计性能时，BMM 的性能始终好于 FMM，且优势先随着训练数据增加而拉大，而后趋于稳定应是因逼近模型性能上限所致。

5.2.3 速度优化方案与进一步思路

课程 PPT 中提供的 trie 树代码使用了字典结构, 直接用于优化原则上不符合实验要求, 因此代码只提交了二分查找的版本，将词典文件中的词读入一个 list 再排序即可。

进一步优化的思路同样基于使用 trie 树。并应该在每一层寻找子节点的部分做改进。trie 树是前缀索引，其搜索子节点的次数为 $O(log_2 n)$. 对于英语来说，每个节点都是单个字母，这意味着每个节点的字节点上限只有 26 个，搜索规模是比较小的。而中文的词普遍较短而单字多，这意味着构建出的 trie 树第一层会有大量的子节点，之后几层子节点数量会急剧减少。在不加优化，不使用哈希的情况下，对中文使用 trie 树只是把一次对整个词表的线性查找变成了一次对所有单字的线性查找 + 几次规模较小的线性查找，且对英语查找次数优势不一定足以弥补首次查找规模的劣势，使得性能改善有限。因此应该对第一层子节点查找的过程做优化，如在寻找第一层的子节点时使用二分查找。

5.3 统计模型的性能分析

对最大概率模型, 基于字的 HMM 模型, 二元基于字的生成式模型与最终的集成模型对全部三个月的训练语料做 10 折交叉验证，得到性能结果表 3.

5.3.1 HMM 性能为什么低?

状态独立性假设是 HMM 的基本假设之一，在本模型中，状态独立性假设意味着对于任意一个 $L \in \{S, B, M, E\}$, 其发射字的概率分布都是相同的。这意味着在建模的过程中我们就舍弃了 n 元文法模型利用的核心信息——字词间的邻接关系。而只靠单个字出现在词中何种位置的概率信息很难充分的对语言建模。

这也就解释了为什么通常使用判别式的由字构词模型需要对当前字开一个窗口抽取特征，抽取特征的过程就将字与字之间的邻接关系纳入了建模范围，可以预见会带来性能的提升。

5.3.2 字生成式模型的分析

字生成式模型在在建模是就同时考虑了有字构词和邻接关系, 且是一个 n 元语言模型, 本次试验中实现的二元模型效果优于一元的最大

模型	准确率	召回率	F1(%)
最大概率	90.93	93.26	92.08
字 HMM	82.81	82.79	82.80
CBGM	94.13	94.35	94.24
集成模型	94.77	94.91	94.84

表 3: 模型性能一览, 使用十折交叉验证

频率分词，也优于是二元但未考虑字邻接信息的 HMM，从理论上讲是合理的。

但是字生成式模型也有其局限之处，其采取 n-gram 的方法利用邻接信息，就意味着对于每个字都只考虑其前面连续 n-1 个字的信息。而加窗口的字判别式模型由于邻接信息不和条件概率绑定，可以同时连续或离散地考虑当前字前后几个字的信息。从这一点来看，使用窗口的字判别式模型会利用到一些字生成式模型不能考虑到的信息，将两者结合会有更好的效果。

5.3.3 基于字模型的合法状态序列

原则上讲，一个句子的字模型下标注并不是四个标签自由组合的序列，而应服从如下正则表达式

$$(S*(BM*E)*)* \tag{10}$$

状态转移矩阵中, a_{SE},a_{SM},a_{MS},a_{BS},a_{BB},a_{EE}, a_{MB},a_{EM} 均应 =0.

但在实际实现数据平滑时并未考虑平滑后会使得模型输出不合法的状态序列，对此采取的手段是修改解读序列的逻辑，使其能够翻译一个不合法的状态序列。其逻辑为维护一个待切分句子 line 的待输出子串 temp，顺序访问第 i 个状态序列，当遇到 E 就将 temp+line[i] 输出; 遇到 S 或 B 时做判断，若 temp 非空则输出 temp，之后 temp=line[i]; 遇到 M 则 temp+line[i] 不输出，若句子结束且 temp 非空则输出 temp.

5.3.4 集成模型的有效性分析

对 CBGM 的错误分词结果做分析，发现当字生成式模型遇到连续的人名/未登录字时性能可能出现滑坡，如下例所示:

模型输出: …/ 著名/ 指挥家/ 陈/ 佐/ 湟/ 、/ 陈/ 燮/ 阳/ 、/ 谭/ 利/ 华/ 分/ 别/ 指/ 挥/ 演/ 奏/ 了/ 一/ 批/ 中/ 外/ 名/ 曲/ , / 京/ 沪/ 两/ 地/ ２００/ 多/ 位/ 音/ 乐/ 家/ 组/ 成/ 的/ 大/ 型/ 乐/ 队/ 以/ 饱/ 满/ 的/ 激/ 情/ 和/ 精/ 湛/ 的/ 技/ 艺/ 为/ 观/ 众/ 奉/ 献/ 了/ 一/ 台/ 高/ 水/ 准/ 的/ 交/ 响/ 音/ 乐/ 会/ 。/

标准答案: …/ 著名/ 指挥家/ 陈/ 佐湟/ 、/ 陈/ 燮阳/ 、/ 谭/ 利华/ 分别/ 指挥/ 演奏/ 了/ 一/ 批/ 中外/ 名曲/ , / 京/ 沪/ 两地/ ２００/ 多/ 位/ 音乐家/ 组成/ 的/ 大型/ 乐队/ 以/ 饱满/ 的/ 激情/ 和/ 精湛/ 的/ 技艺/ 为/ 观众/ 奉献/ 了/ 一/ 台/ 高/ 水准/ 的/ 交响音乐会/ 。/

一个简单的判断分词结果是否出现滑坡的方法是计算切分词数和句子字数之比，当比值过高时认为该分词结果不可信任，重新采用 BMM 分词。经试验发现比较理想的阈值应设定在 0.75 左右。

6 结论

本次试验对中文分词任务做了简单的梳理，介绍了 FMM，BMM, 最大概率，基于字的 HMM，基于字的生成式模型和一个集成模型的原理和算法实现思路，给出了这些模型各自的性能。并具体讨论了构建分词系统时词典构建与正预处理的关系, 比对了 BMM 和 FMM 之间的性能差异，分析了基于字模型对语料信息的利用程度和数据平滑时可能引起的输出序列不合法问题。

局限和不足

在构建基于字的模型时没有强制要求字的状态序列是合法的，而是在输出时按一定规则调整结果，在一定程度上影响了模型性能，且不利于进行错误的分析与分类。

一些数据平滑手段没有确保平滑后全体概率和仍为 1，在理论上不够严谨。

参考文献

Kun Wang, Chengqing Zong, and Keh-Yih Su. 2009. Which is more suitable for Chinese word segmentation, the generative model or the discriminative one? In *Proceedings of the 23rd Pacific Asia Conference on Language, Information and Computation, Volume 2*, pages 827–834, Hong Kong. City University of Hong Kong.

宗成庆. 2008. 统计自然语言处理（第 2 版）. 清华大学出版社.

A 代码提交有关说明

需要使用正则库 re, 运行 HMM 代码需要安装 numpy

运行 main.py 获取测试用的 seg_LM.txt,3.5 优先使用 CBGM 提交；3.6 提交的两个模型分由 CBGM_poct 和 CBGM_poct2 调用

unigram.py: 最大概率分词代码，可修改 main 函数中文件名参数单独运行

DAG.py：ppt 中提供的有关代码

HMMC2.py: 基于字的 HMM 代码，可单独运行；此外这个文件中还有词——标签，分词标签结果——词的转换函数

CBGM.py：字生成式模型代码，可单独运行

PostProcess.py: 集成模型的代码和后处理的几个函数，可单独运行

regex_preprocess.py: 与正则有关的各种代码

BMM.py 重新提交一版，增加了用于集成模型的类 PortableBMM

Evaluate.py: 分词结果评价函数 evaluation()，十折交叉验证函数 ten_fold()，将原始语料转化成分词结果和测试文件的函数 get_answer()，用于将指定文件分成 10 折的函数 partition()

all 开头的 txt 文件均为三个月训练语料的不同版本（原始，测试，分词结果）

dic.txt：字典

Parts 文件夹：切分好的十折语料，默认用的是全体，需要用第一月的十折需要到 Evaluate.py 改对应文件名（在前面 +01）

PartDics 文件夹：存放之前几个模型十折的运行结果