基于关联规则的自动构词算法研究

王鉴全 季绍波
(大连理工大学管理经济学部 大连 116023)

摘 要 词语是中文文本的基本元素,汉语语言模型在中文文本挖掘中起关键作用。中文文本挖掘是高维度的数据处理技术,挖掘算法对维度的大小比较敏感,因此挖掘效果依赖于词库的质量。另外,现存的汉语语言模型一般都是基于统计的,比如N-gram语言模型以及各种改进模型都具有较高的计算复杂度。为降低语言模型的计算复杂度、提高词库的质量和构词效率,借鉴关联规则理论对中文词语进行定义,在此基础上构建Auto-word自动构词算法。该算法可以从大量中文语料库中动态地构造词表,并以此为基础进行中文文本挖掘工作。最后通过实验证明了提出的自动构词算法的有效性。

关键词 自动构词,统计语言模型,关联规则,最长公共子序列,文本分类
中图法分类号 TP301.6 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.11.049

Research and Application on Auto-word Building

WANG Jian-quan JI Shao-bo
(Faculty of Management and Economics, Dalian University of Technology, Dalian 116023, China)

Abstract Words are the basic elements of Chinese text, and Chinese language model plays a key role in Chinese text mining. Text classification is a data mining technology with high dimensions and most of the classifying algorithms are sensitive to the dimensions. As a result, the classification depends on the quantity of vocabularies. Besides, most of current Chinese language models are based on statistical theory, such as N-gram model and other improved models. However, these statistical models are disadvantaged with computational complexity. In order to improve the quantity and efficiency, this paper gave Chinese words a new definition based on association rules, and proposed the Auto-word algorithm, by which a word vocabulary is constructed automatically and used for Chinese text mining. Finally, the efficiency of the Auto-word algorithm was proved by experiment.

Keywords Constructing words automatically, Statistical language model, Association rules, Longest common subsequence, Text classification

1 引言

在中文信息处理领域,汉语语言模型发挥了至关重要的作用,已经成功应用到文本挖掘[1]、语音识别[2]、机器翻译[3]、智能推荐、搜索引擎等多个领域当中。特别是在中文文本挖掘领域,高质量的语言模型可以大幅提高文本挖掘的准确度。

关于中文文本挖掘研究,一般都是基于机器学习的分类模型,例如决策树、神经网络、支持向量机等[4],这些模型先是通过一个正确的样本集训练出一个有效的规则,然后通过这个规则去检验未知分类的文本信息[5]。在实际应用中,无论使用哪种分类模型,通过中文分词算法提取出文本的特征词都是一个必需的前提工作。现存的中文分词算法有正向最大匹配算法、逆向最大匹配算法、最少切分算法以及基于统计语言模型的分词算法,各种算法都有一定的优缺点[6]。但在提取文本特征准确性上,算法之间没有太大区别,更多地依赖于分词词表或语言模型的准确性。对于一些文本来源比较稳定的文本挖掘问题,可以使用一个统一大词表(比如北大词表[7])来实现中文分词并提取文本特征。但大多情况下,待处理的文本挖掘问题都具有特定的领域背景,或者不确定的信息来源,使用统一的词表将降低准确度和执行效率。这时更需要一个动态的词表或者能直接动态地获取文本特征。

现存的汉语语言模型大都是基于统计的,其中最常用的模型为N-gram语言模型和改进模型[8]。N-gram语言模型基于一个假设,即第N个字的出现只与前$N-1$个字相关,并在此假设基础上计算任意一个文字序列w出现在文本中的概率$p(w)=p(z_1)p(z_2|z_1)\cdots p(z_N|z_1z_2\cdots z_{N-1})$。其中每个条件概率都来自训练样本的统计结果$p(z_i|z_1z_2\cdots z_{i-1})=\frac{f(z_1z_2\cdots z_{i-1}z_i)}{f(z_1z_2\cdots z_{i-1})}$,$f(x)$是序列$x$在训练文本中出现的频率[8]。从N-gram模型公式可看出,当N较大时,计算量与存储量大幅增加,同时会出现数据稀疏问题。实际应用时,将折中考虑计算精度和计算效率,选择适当的N值,如$N=2$时N-gram为二元模型(bi-gram);$N=3$时N-gram为三元模型

到稿日期:2014-01-02 返修日期:2014-03-24
王鉴全(1971—),男,博士生,主要研究方向为文本挖掘,E-mail:wangjq26@126.com;季绍波(1960—),男,博士,教授,主要研究方向为信息系统管理、文本挖掘。

(tri-gram)，当 N 取值稍大时也可以通过补偿平滑技术改进模型。但一般情况下，计算精度与计算效率不可两全。

本文在关联规则理论的基础上提出一种自动构词算法，该算法不受 N 值限制，可以从大量文本中自动快速地获取文本中常用的词语，并构成词库。在对文本进行分类时，从分类文本中提取该词库，并用该词库提取文本特征，则获取的特征将更具有代表性，分类结果更加准确。

2 基于关联规则理论的词语定义

字是组成文本的最小元素，词语是字为表示特定概念按照一定顺序组合在一起所形成的文字序列。设所有字的集合为 $Z=\{z_1,z_2,z_3,\cdots,z_n\}$，则一个词 w_i 可以表示为 $z_{i1}z_{i2}z_{i3}\cdots z_{ik}$，其中 $z_{ij}\in Z$，并记词的集合为 $W=\{w_1,w_2,w_3,\cdots,w_m\}$。对于比较复杂的词语也可以通过词或字之间相互组合来表示，比如 $w_1=z_1z_2$，$w_2=z_3z_4z_5$，$w_3=z_1z_2z_3z_4z_5$，则 $w_3=w_1w_2$，$w_3=z_1z_2w_2$，$w_3=w_1z_3z_4z_5$。

定义 1(文字序列的支持度) 设 $w_i=z_{i1}z_{i2}z_{i3}\cdots z_{ik}$ 是可以从文本集合中提取的一个文字序列，定义该文字序列在所有文本集中出现的频率为该文字序列的支持度，即支持度 $Support(w_i)=\dfrac{freq(w_i)}{|T|}$，其中 $|T|$ 是文本集 T 的文本总数量，$freq(w_i)$ 是文本集 T 中出现文字序列 w_i 的文本数量。

根据定义可知，文字序列的支持度就是该序列在一个文本中出现的概率，即 $Support(w_i)=P(w_i)$。根据贝叶斯概率公式，文字序列的支持度又可表示为：

$$\begin{aligned}Support(w_i)&=P(z_{i1}z_{i2}z_{i3}\cdots z_{ik})\\&=P(z_{i1}z_{i2}\cdots z_{ij})*P(z_{ij+1}z_{ij+2}\cdots z_{ik}\mid z_{i1}z_{i2}\cdots z_{ij})\end{aligned}$$

记子序列 $w_i^{1,j}=z_{i1}z_{i2}\cdots z_{ij}$，子序列 $w_i^{j+1,k}=z_{ij+1}z_{ij+2}\cdots z_{ik}$，则有 $P(w_i)=P(w_i^{1,j})*P(w_i^{j+1,k}\mid w_i^{1,j})$，或 $P(w_i^{j+1,k}\mid w_i^{1,j})=\dfrac{P(w_i)}{P(w_i^{1,j})}=\dfrac{Support(w_i)}{Support(w_i^{1,j})}$。

定义 2(有效拆分) 对于任意一个文字序列 $w_i=z_{i1}z_{i2}z_{i3}\cdots z_{ik}$，可以将其拆分成多个子序列。如果存在一个拆分：

$$\begin{aligned}w_i&=z_{i1}z_{i2}z_{i3}\cdots z_{ik}\\&=z_{i1}z_{i2}\cdots z_{ij}z_{ij+1}z_{ij+2}\cdots z_{ik}\\&=w_i^{1,j}w_i^{j+1,k}\end{aligned}$$

并且 $w_i^{1,j}$、$w_i^{j+1,k}$ 都是一个词语，则称该拆分为文字序列 w_i 的一个有效拆分。

根据定义可知，可以进行有效拆分的文字序列实际上就是一个由两个词语组成的复合短语，而且对于某些复合短语，存在不止一种有效拆分。例如文字序列“北京人民大会堂”可以拆分成“北京人民”和“大会堂”，同时也可拆分成“北京”和“人民大会堂”。事实上，一个文字序列之所以能称为一个词语，主要是因为该文字序列能独立地表达一种意思。如果该文字序列能拆分成两个更具有表达力的词语，那么该文字序列作为一个词语的意义将会降低。

定义 3(文字序列的置信度) 文字序列的置信度是指该文字序列称为词语的信任度，如果该文字序列不存在有效拆分，则该文字序列的置信度为 1；否则，该文字序列在一个或多个有效拆分中都存在一个基于前子序列的条件概率，取最小的条件概率为该文字序列的置信度。即

$$Confidence(w_i)=\begin{cases}\min\limits_{1<j<k}P(w_i^{j+1,k}\mid w_i^{1,j}), & \text{存在有效拆分}\\1, & \text{不存在有效拆分}\end{cases}$$

根据定义，对于不能进行有效拆分的文字序列其置信度为 1，不需要计算。对于能进行有效拆分的文字序列，其表达式可进一步表示为：

$$\begin{aligned}Confidence(w_i)&=\min_{1<j<k}P(w_i^{j+1,k}\mid w_i^{1,j})\\&=\min_{1<j<k}\frac{Support(w_i)}{Support(w_i^{1,j})}\end{aligned}$$

根据关联规则理论，一个关联规则能称为强关联规则，应同时满足最小支持度阈值和最小置信度这两个条件[9]。本文将强关联规则作为衡量文字序列是否可称为词语的依据，根据关联规则理论对词语做如下定义。

定义 4 如果一个文字序列能同时满足最小支持度和最小置信度两个条件，那么该文字序列可作为一个独立的词语存在。

最小支持度与最小置信度是文字序列成为词语的充分必要条件。在现实文本中，存在大量高频短语，但如果某高频短语能拆分成更加高频的词语，那么使用该高频短语作为独立词语的意义将被消弱。相反，如果一些不可拆分的短语在文本中经常出现，那么它们将很有必要作为一个词语来使用。

3 Auto-word 自动构词算法

挖掘关联规则的通用算法是一种计算量比较复杂的算法，比如 Apriori 算法[10]、Trie 字典树算法[11]等。由于词语本质上就是连续的多个字组成的序列，在词语中字与字之间存在严格的顺序关系。本文在 Trie 字典树算法的基础上，提出了一种专门处理中文本文词语挖掘的算法，称为 Auto-word 算法。

与 Trie 字典树算法类似，Auto-word 算法挖掘关联规则的过程分为两个阶段：第一阶段，先找到满足最小支持度的频繁集，并构造 Trie 树；第二阶段，由频繁集产生同时满足最小支持度和最小置信度的强关联规则，即发现词语。但与 Trie 字典树不同的是，在第一阶段，Auto-word 算法首先找出可能的所有阶的频繁集，并将频繁集按照字序列中字的顺序存储到字典树中，而 Trie 字典树算法是按照项目出现的频率大小顺序创建字典树；第二阶段，Auto-word 算法按照宽度优先的规则逐层遍历字典树，判断文字序列是否能构成强关联规则，将构成强关联规则的文字序列确认为词语并保存在字典树中，否则删除该序列，而 Trie 字典树算法是按照深度优先的规则遍历字典树。

3.1 提取文字序列频繁集

中文文本的最小单位是文字，由于词语是连续的文字组成的，可根据最长公共子序列的算法(Longest Common Subsequence，LCS)获取中文语句中最长的文字序列[12]。本文对 LCS 算法稍做改动即可获取中文语句中所有可能成为词语的文字子序列，算法步骤总共分 3 步。

第一步 构造比较矩阵 Matrix。设两个短句分别为 A、B，A 中字的数量为 m，B 中字的数量为 n。比较矩阵就是一个 $m*n$ 的矩阵，元素取值 0 或 1。A 中第 i 个字与 B 中第 j 个字比较，如果相等，元素(i,j)的值为 1，否则为 0，即：

$$Matrix=\begin{cases}1, & A_i=B_j\\0, & A_i\neq B_j\end{cases}$$

第二步 寻找长度大于 1 的公共子序列。子序列是连续的元素，从比较矩阵中找出公共子序列，就是找出元素连续相等的元素序列，从矩阵来看就是对角线上连续为 1 的元素序

列。所以寻找长度大于 1 的公共子序列的伪代码算法如图 1 所示。

```
Algorithm: Find_CS(){
1. global Matrix; //全局变量,比较矩阵
2. i=1;
3. while( i<m)
4. {
5.     j=1;
6.     while(j<n)
7.     {
8.       if(Matrix[i][j]==1)
9.       {
10.          Matrix[i][j] = Matrix[i-1][j-1] + 1;
11.      }
12.      j++;
13.    }
14.    i++;
15.  }
16. }
```

图 1　寻找长度大于 1 的公共子序列

第三步　提取公共子序列。为避免对文字序列重复统计,对于每个长度大于 2 的公共子序列,其子序列将不再作为一个独立的子序列提出。例如,如果"上海人"是两个句子中的一个公共子序列,那么其子序列"上海"将不再重复作为一个公共子序列。算法如图 2 所示。

```
Algorithm: Get_res(){
1. global Matrix; //全局变量,比较矩阵
2. Results=[]
3. i=m;
4. while(i->1)
5. {
6.   j=n;
7.   while(j->1)
8.   {
9.     if(Matrix[i][j]>1)
10.    {
11.      Results. append(M. substr(i+1- Matrix[i][j],
             Matrix[i][j]));
12.      ii=i-1;
13.      jj=j-1;
14.      while(Matrix[ii][jj]>0)
15.      {
16.        Matrix[ii--][jj--]=0;
17.      }
18.    }
19.  }
20. }
```

图 2　提取长度大于 1 的公共子序列

在对大量的文本进行断句并提取所有可能的公共子序列的同时,统计出每个公共子序列在中文文档集合中出现的频率。根据预设的最小支持度筛选出频繁的文字序列,构建频繁集。

3.2　构造字典树

字典树在文本挖掘中经常使用,它将文字序列放在一个树结构中,每个字对应树中的一个节点;前后两个字在树结构中构成父子节点关系,并通过 hash 表存储每个节点所拥有的孩子节点;一个独立的子序列在字典树中对应一个从根节点到某个节点的链路,并在序列末尾对应节点上标记该序列出现的频率。通过字典树可以快速查找某个文字序列是否出现在集合中。表 1 是一个文字序列集合,图 3 为该序列集合对应的字典树。当文字序列集合增大时,通过字典树查找某个文字序列的速度只与文字序列的长度有关。

表 1　文字序列集合示例

文字序列	频数
中国	16
中国人	9
中国人民	5
中国银行	7
中间	5

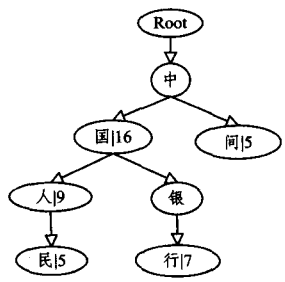

图 3　文字序列集合对应的字典树

3.3　挖掘词语

与 Trie 字典树算法不同,Auto-word 算法在挖掘强关联规则的过程中是逐层处理的,首先遍历字典树中由两个字组成的序列,根据定义 3 中的置信度计算公式计算每个序列的置信度,判断是否构成强关联规则,若构成强关联规则则选定为词语,否则删除该子序列;然后依次遍历 3 个和更多个字组成的序列的置信度。在分阶挖掘词语时,遍历字典树的过程使用宽度优先遍历算法(visit 算法),算法描述如图 4 所示。

```
Algorithm: visit()
{
  Global Dicttree; //全局变量,字典树
  queue=[]; //创建空队列
  For child in Dicttree. root. children{
      Queue. append(child); //将节点压入队列尾部
  }
  While( ! queue. empty() ){ //依次处理每个节点
      node = queue. popfirst();//从队列头部获取节点
      Sequence=get_seq(Dicttree,node);//根据当前节点获取子序列
      If(judge(Dicttree,sequence)){//判断序列是否构成词语
        Node. is_word=True; //将该节点标识为词语
      }
      For child in node. children{
          queue. append(child);
      }
  }
}
```

图 4　通过宽度优先遍历字典树挖掘词语

在图 4 中的算法中,用到了 get_seq 函数以及 judge 函数,其中 get_seq 函数通过寻找根节点到当前节点的链路来获取当前节点对应的文字序列;judge 函数计算文字序列的置信度并根据定义 4 判断该文字序列是否构成词语,其算法描述如图 5 所示。

```
Algorithm: judge(Dicttree,sequence)
{
  idx=0;
  confidence=1; //置信度初始化为 1;
  While(idx<sequence.length){
    idx++;
    s1= sequence.sub(0,i); //获取前子序列
    s2= sequence.sub(i,sequence.length);//获取后子序列
    If(is_word(s1)&&is_word(s2)){ //构成有效划分
      cf= sequence.frequence/s1.frequence; //计算置信度
      confidence=min(cf,confidence); //置信度取小
    }
  }
  return confidence>MIN_CONFIDENCE; //是否满足最小置信度
}
```

图 5 判断文字序列是否构成词语的算法描述

经过 visit 算法遍历字典树之后，字典树中剩余的文字序列均构成词语，再次遍历字典树就可获取所有挖掘出来的词语。

3.4 实验

语言模型的有效性 F 一般通过考察准确率 P 和召回率 R 来衡量[8]。准确率就是抽取正确的词语数量与所有抽取的词语数量的比值[13]；召回率是自动抽取并出现在人工词表中的词语数量与人工词表数量的比值。准确率和召回率的计算均基于随机样本抽样方式，语言模型的有效性 F 最终由准确率 P 和召回率 R 共同计算得出，计算公式为：

$$F=\frac{1}{\theta\frac{1}{P}+(1-\theta)\frac{1}{R}}$$

其中 θ 为常数，表示准确率在评价模型有效性中的重要程度，一般取 $\theta=0.5$。

实验一 相同阈值情况下与 N-gram 模型的对比。为验证本文提出的算法的有效性，实验样本取自中文新闻网(chinanews.com)的 10 万篇中文网页，根据经验，Auto-word 算法最小支持度取值 0.001，最小置信度取值 0.05，同时使用分别基于二元模型和三元模型的 N-gram 算法，词频阈取值 10。在同样的硬件环境中分别运行 3 种模型，并记录各自抽取的词语数量和运行时间，同时采用随机抽样的方式统计构词词语的有效性 F，结果对比如表 2 所列。

表 2 本文模型与二元、三元模型比较

模型	词语数量	运行时间(秒)	准确率	召回率 F	有效性
二元语言模型	21534	816	0.654	0.696	0.674
三元语言模型	18412	2141	0.739	0.645	0.689
关联规则模型	28124	965	0.751	0.675	0.711

由表 2 可以看出，与常用的 N-gram 语言模型相比，本文提出的基于关联规则的自动构词语言模型更有效，获取的词语更全面，同时也能保持较高的运行效率。

实验二 不同最小支持度情况下 Auto-word 算法运行结果的对比。关联规则挖掘结果对最小支持度的取值比较敏感，一般地，当最小支持度较大时，频繁集较小，准确率上升，但获取的词语数量减少，召回率下降。本文通过设置不同的最小支持度，对中文新闻网(chinanews.com)的 10 万篇中文网页进行词语挖掘，实验结果如表 3 所列。

表 3 不同最小支持度下词语挖掘结果比较

最小支持度	词语数量	运行时间(秒)	准确率	召回率 F	有效性
0.0005	43789	3701	0.446	0.778	0.567
0.001	28124	965	0.751	0.675	0.711
0.002	11673	508	0.803	0.411	0.544
0.004	3925	135	0.894	0.127	0.222

由表 3 可以看出，选择不同的最小支持度，Auto-word 算法挖掘词语的效果差别很大。最小支持度太小，挖掘的词语数量增加，运行时间增加，准确率下降，召回率增加；最小支持度太大，有效性降低，挖掘词语数量减少，运行时间减少，准确率上升，召回率下降，有效性也降低。由于不同领域的样本集具有不同的词语结构，在从特定领域的文本集中提取词语时，应适度调整最小支持度，以获取最优的词语挖掘效果。

结束语 本文从中文文本挖掘使用的词库出发，基于关联规则理论对中文词语进行定义，并在此基础上提出了 Auto-word 自动构词算法。该算法在处理各种中文文本挖掘问题时，可以降低文本特征空间的维度，提高词语的质量。实验证明，本文提出的语言模型比常用的 N-gram 语言模型更有效。但自动构词算法是一项比较耗时的工作，当待挖掘的文本数量较大或设定的最小支持度和最小置信度较小时，需要提取的词语量较大，自动构词消耗的时间也会增加，Auto-word 算法同 N-gram 一样将面临运行效率问题。所以如何提高构词速度是我们下一步的主要工作，一方面在词语提取算法和短句选择上进行优化，另一方面可以采用分布式的并行计算技术来提高自动构词速度[14]。

参 考 文 献

[1] 苏菲，王丹力，戴国忠. 基于标记的规则统计模型与未登录词识别算法[J]. 计算机工程与应用，2004(15)：43-45

[2] 李伟，吴及，吕萍. 基于前后向语言模型的语音识别词图生成算法[J]. 计算机应用，2010，30(10)：2563-2566

[3] 刘群. 统计机器翻译综述[J]. 中文信息学报，2003，17(4)：1-12

[4] 张苗，张德贤. 多类支持向量机文本分类方法[J]. 计算机技术与发展，2008，18(3)：139-141

[5] 刘红岩，陈剑，陈国青. 数据挖掘中的数据分类算法综述[J]. 清华大学学报：自然科学版，2002，42(6)：727-730

[6] 张启宇，朱玲，张雅萍. 中文分词算法研究综述[J]. 情报探索，2008(11)：53-56

[7] 俞士汶，朱学锋，王惠，等. 现代汉语语法信息词典详解[M]. 北京：清华大学出版社，1998

[8] 肖镜辉，刘秉权，王晓龙. 面向汉语建模的自适应词表生成算法[J]. 自动化学报，2008，24(1)：40-47

[9] 刘君强，孙晓莹，潘云鹤. 关联规则挖掘技术研究的新进展[J]. 计算机科学，2004，31(1)：110-113

[10] Agrawal R, Srikant R. Fast algorithms for mining association rules[C] // Proc. 20th Int. Conf. Very Large Data Bases (VLDB). 1994, 1215: 487-499

[11] Amir A, Feldman R, Kashi R. A new and versatile method for association generation [M] // Principles of Data Mining and Knowledge Discovery. Springer Berlin Heidelberg, 1997: 221-231

[12] 王映龙，杨炳儒，宋泽锋，等. 基因序列相似程度的 LCS 算法研究[J]. 计算机工程与应用，2007，41(31)：45-47

[13] Fung P. Extracting key terms from Chinese and Japanese texts [J]. Computer Processing of Oriental Languages, 1998, 12(1): 99-121

[14] 程苗，陈华平. 基于 Hadoop 的 Web 日志挖掘[J]. 计算机工程，2011，37(11)：37-39