一种基于后缀数组聚类(SAC)的中文垃圾邮件过滤方法

李翔鹰[1] 陈 钟[1] 唐礼勇[1] 李 欣[2]
(北京大学计算机科学技术系 北京 100871)[1]
(辽宁工程技术大学电子信息与工程系 辽宁阜新 123000)[2]

摘 要 贝叶斯算法在垃圾邮件过滤中应用广泛,但在中文垃圾邮件过滤中性能较低。本文通过聚类的思想,提出一种基于后缀数组聚类(SAC)的中文邮件特征项抽取方法,并给出了不同特征项抽取方法下贝叶斯算法的中文垃圾邮件过滤实验数据对比。实验表明,该方法显著提高了中文垃圾邮件的过滤性能。

关键词 朴素贝叶斯,垃圾邮件过滤,后缀数组

A Method of Chinese Spam Filtering Based on Suffix Array Clustering (SAC)

LI Xiang-Ying[1] CHEN Zhong[1] TANG Li-Yong[1] LI Xin[2]
(Department of Computer Science and Technology, Peking University, Beijing 100871)[1]
(Electronics and Information Engineering Department, Liaoning Technical University, Liaoning 123000)[2]

Abstract The naive-bayes algorithm has widely been applied to spam filtering. However, it has unsatisfactory performance in Chinese email filtering. Using clutering, this paper proposes a suffix array clustering based token extraction method for Chinese email, named SAC. It also shows the different filtering results of bayes under different token extraction methods. The experiments domenstrate the improvement of filtering performance of the method for Chinese spam.

Keywords Naïve-bayes, Spam filtering, Suffix array clustering

1 前言

随着因特网的普及,垃圾邮件日益严重。目前,通过文本分类的方法来解决基于内容的垃圾邮件过滤问题成为国内外研究的热点。其中,贝叶斯算法由于其分类效果好、训练和分类时间短,是垃圾邮件过滤中常用的方法之一。典型的有:朴素贝叶斯算法[1]以及 Paul Graham 提出的基于贝叶斯规则的垃圾邮件过滤算法[2,3](下文简称 PG 贝叶斯算法)。PG 贝叶斯算法较朴素贝叶斯算法更为简单、高效,因此在垃圾邮件过滤中得到广泛应用[4]。

上述贝叶斯算法在基于内容的英文邮件过滤上表现出良好的性能,但是还没有实验表明这些算法也能很好地运用在中文垃圾邮件过滤上。

构成邮件内容的文档有两个基本特征:组成文档的所有字词符号和这些字词间的排列顺序。贝叶斯算法首先以字或词为特征项,将文档表示为空间向量,通过学习已标注类别的训练样本获得特征项是"垃圾特征"的概率,进而对待分类邮件依据其特征项进行判断。因此,特征项的选取是算法的基础并直接影响分类算法的分类结果。英文邮件过滤中,通常选取单词作为特征项;而中文不像英文那样有明确的分隔符(例如空格或定界符等),文[5]采用切词方法并辅以动态自学习汉语词库系统用于中文特征项抽取后,应用朴素贝叶斯算法过滤垃圾邮件;文[6]使用 N-gram 对中文文档进行自动分词,得到特征项表示,通过朴素贝叶斯算法过滤垃圾邮件。上述文献中均未见不同特征项抽取方法下实验数据对比,因此有必要对其进一步研究,解决贝叶斯算法对中文邮件过滤的适应性。

本文提出了一种基于后缀数组聚类的中文垃圾邮件特征项抽取方法 SAC(Suffix Array Clustering),它充分利用数据的自然结构和特点,通过聚类将数据划分为不同的"簇",以聚类标识作为特征项。将其应用于 PG 贝叶斯算法,显著提高了中文垃圾邮件过滤性能。

本文的第 2 部分提出中文邮件特征项抽取方法 SAC,第 3 部分描述了 SAC 的设计与实现,第 4 部分给出不同特征项抽取方法下贝叶斯算法的中文邮件过滤实验数据对比。最后是结论和下一步工作。

2 SAC:中文邮件特征项抽取方法

2.1 相关特征项抽取方法分析

通常,中文特征项的选取方法有:词典方法、语言学方法和统计的方法。词典方法即中文切词的方法,仅把文档看作是词语的集合,忽略了它们之间排列顺序所表达的信息;此外,该方法依赖于词典,无法适应垃圾邮件中不断出现的新词汇,特别是垃圾邮件制造者为了避免邮件被过滤而故意拼写错误的词汇。

语言学的方法通过自然语言处理技术对特征项进行高层次的抽取,需要借助已经建立的词典和语料库,从中学习规则,发现语言现象。其规则维护困难。

统计的方法即"词袋"抓取法,按照统计学的方法选取一定长度的短语。短语的长度为 2,3,4 或 5。这种方法考虑了

李翔鹰 硕士研究生,主要研究方向:网络与信息安全;陈 钟 博士,教授,主要研究方向:网络与信息安全、软件工程;唐礼勇 博士,副教授,主要研究方向:网络与信息安全;李 欣 硕士研究生,主要研究方向:网络与信息安全。

词语之间的顺序信息，但仍然按照预先定义的长度值截取特征项。

2.2 SAC的基本思想

聚类是把大量的数据样本聚集成 k 个类，使同一类中样本的相似度最大，而不同类间的样本相似度最小。很自然地，我们考虑利用聚类方法从数据自身特点中抽取特征项，通过聚类后的类别标识得到“抱团”特征项数据表示，从而实现特征项的抽取。同时，在垃圾邮件过滤中还应考虑到垃圾邮件是动态变化的信息流，不断会有新的“垃圾”特征出现。因此，特征项的抽取必须是增量式的，以满足特征项集合重构的要求，即聚类的类别个数是增量变化的。

后缀树是一种广泛用于串匹配和串查询的数据结构。O. Zamir 和 O. Etzioni 采用后缀树聚类(Suffix Tree Clustering)的算法(下面简称 STC)研制出一个自动聚类系统 Grouper，可针对用户的查询要求对网页内容进行增量聚类[7]。

下面给出后缀树的相关定义[8]，并假定字串 S_i 由不同单词组成。

定义 1 由 n 个长度为 m_n 的字串 S_i 的集合组成的平凡后缀树 T 是一棵有根的有向树，该树有 $\sum m_n$ 个叶子，每个叶子节点由 2 元组 (K,L) 表示。K 代表串的标识，取值从 1 到 n；L 代表后缀串在其所在串中的起始位置，取值从 1 到 m_i。除根节点以外的内部节点，至少有两个孩子节点，每个边由 S_i 的非空子串标注。任何起始于一个节点的边开始于不同的单词。对任意叶子节点 (i,j)，从根到叶子 (i,j) 的边的串联标识了串 S_i 的从起始位置 j 开始的后缀：$S_i[j,\cdots,m_i]$。

图 1 是一个由 3 个字串组成的平凡后缀树的例子：“cat ate cheese”，“mouse ate cheese too”和“cat ate mouse too”。后缀树的内部节点表示为圆圈，并从 a 到 f 标注；叶子节点表示为矩形。矩形中的两个数字分别对应上述定义中的 2 元组：第一个数字表示串标识，第二个数字表示在该串中后缀串的起始位置。每个串都以一个不在串语言集合的终止符结束(如 $，没有在图中标出)。

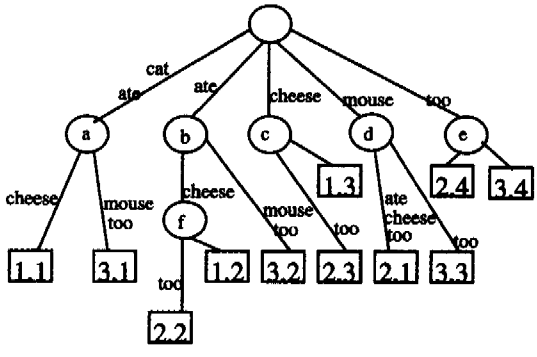

图 1 平凡后缀树示意图

STC 将那些共同出现次数多而且长度长的短语作为聚类标识来揭示聚类的内容。类似地，我们可以在文档的特征项空间中进行聚类，用聚类后的类别标识来反映句子中共同出现频率高且长度较长的短语，并作为特征项。这样，特征项的抽取就是数据驱动的，即特征项的选取依赖于其对文档内容的贡献度，而特征项的长度仅由特征项本身决定。

3 SAC方法的设计与实现

SAC 方法的基本步骤是：

(1)对中文邮件文档的内容以句子为单位，扩展每个句子的所有后缀串，得到所有后缀串的共同前缀，构成候选特征项集合 V_0；以特征项的出现频率和特征项的长度为标准，从候选特征项集合 V_0 抽取特征项，生成特征项集合 V；

(2)从候选特征项集合 V_0 中选取长度长而在(2)中没有选取的后缀串，补充加入特征项集合 V。

下面详细描述 SAC 的实现。

3.1 抽取特征项

STC 通过直接构造后缀树的方法[9]进行聚类，而后缀数组是一种比后缀树节省空间的数据结构。与后缀树相同，该数据结构能高效地实现串匹配和模式发现[10]。文[11]在基因比较问题中，对后缀树和后缀数组做了深入的分析，并得出结论：每一个以后缀树作为数据结构的算法都可以借助一些数据表的帮助，在同样的时间代价上以后缀数组的数据结构实现。其核心思想是构造了一棵概念上的最长共同子串内部树(lcp-interval)。图 2 和图 3 分别是串 S=acaaacatat $ 的相关后缀数组表和 lcp-interval 树。

相关数据表定义如下：

定义 2(suftab 数组) 数组元素是串 S$ 的 $n+1$ 个后缀串，且各后缀串按照字典序升序排序。

定义 3(lcptab 数组) 记录相邻 suftab 数组的元素之间的最长共同前缀的长度。定义 lcptab[0]=0，lcptab[i]为 suftab[$i-1$]和 suftab[i]之间的共同前缀($1\leqslant i\leqslant n$)。由于串以 $ 结尾，lcptab[n]=0。

i	0	1	2	3	4	5	6	7	8	9	10
lcptab	0	2	1	3	1	2	0	2	0	1	0
suftab	aaacatat$	aacatat$	acaaacatat$	acatat$	atat$	at$	caaacatat$	catat$	tat$	t$	$

图 2 串 S=acaaacatat $ 的相关后缀数组表

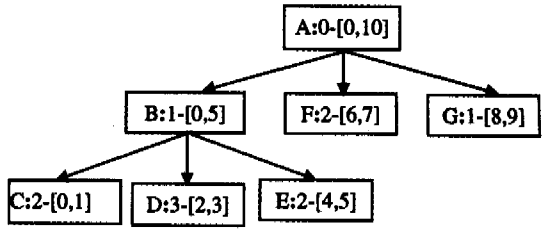

图 3 串 S=acaaacatat $ 的 lcp-interval 树

区间间隔为[i，j]的长度为 k 的最长公共子串可以简写为 k—[i，j]。从图 3 可以看出，节点 B 表示从位置 0 到位置 5 的最长公共前缀的长度是 1，而其下三个孩子节点分别表示：从位置 0 到位置 1 最大公共前缀长度为 2；从位置 2 到位置 3 最大公共前缀长度为 3；从位置 4 到位置 5 最大公共前缀长度为 2。将 suftab 数组中的单个字符扩展为独立的中文汉字，利用这种方法可以求出满足条件的特征项。

SAC 进行特征项抽取时分为 3 个步骤：

(1)把一篇文档看作是 N 个字的集合，构造其按照字典排序的 suftab 位置数组。

(2)求所有位置数组中前向相邻的任两个数组之间的最大共同前缀，即生成 lcptab 数组。

(3)遍历 lcptab 数组,生成(概念上)lcp-interval 树,求出所有子串最大公共前缀。

遍历算法如下:

```
Begin
1 : lastInterval : =⊥//⊥代表空节点
2 : push((0,0,⊥,[]))
3 : for i : =1 to n do
4 :   lb : =i-1
5 :   while lcptab[i]<top. lcp do
6 :         top. rb : =i-1
7 :         lastInterval : =pop
8 :         process(lastInterval)
          //对满足条件的 lcpInterval 节点处理
9 :         lb : =lastInterval. lb
10 :        if lcptab[i]<=top. lcp then
11 :        top. childList : =add(top. childList,lastInterval)
12 :        lastInterval : =⊥
13 :      endif
14 :   enddo
15 : if lcptab[i]>top. lcp then
16 :      if lastInterval<>⊥ then
17 :      push((lcptab[i],lb,⊥,[lastInter val]))
18 :          lastInterval : =⊥
19 :    else push((lcptab[i],lb,⊥ : [lastInter val]))
20 :        endif
21 : endof
End
```

遍历算法的输入是 lcptab 数组(相邻 suftab 数组的元素之间的最长公共前缀的长度),输出是 lastInterval 节点((lcp,lb,rb,childList))。其中,lcp 是区间间隔为[lb,rb]的最大公共前缀的出现次数,lb 和 rb 分别是最大公共前缀在 lcptab 索引左边界和右边界的整数表示,决定了最大公共前缀的内容及长度;childList 是孩子节点的列表。因此,算法第 8 行的函数 Process(lastInterval)中,根据 lastInterval 信息((lcp,lb,rb,childList))将特征项的长度和特征项的出现次数的乘积为衡量标准,抽取出公共前缀长度长。而出现次数多的短语作为特征项,构成特征项集合 V。

3.2 生成特征项集合

考虑到垃圾邮件的内容在一定的时间内有局部重复性和相似性的特点,从文档候选特征项集合 V_0 中选取共同出现次数不够多而长度比较长的字串,如果不在特征项集合 V 中,则加入之。这样做可以进一步增强 SAC 对特征项长度的倚重,有利于捕获到句子或片段相似的垃圾邮件。

3.3 SAC 方法分析

SAC 方法中,生成字典排序的 suftab 位置数组的时间代价为 $O(N*\log_2 N)$,其余步骤中均为 $O(N)$。因此,SAC 的时间代价为 $O(N*\log_2 N)$。

另外,在增量式聚类的应用中,每加入一个句子后缀串,其对数据集合的影响是局部的:如新的后缀串插入后,后缀数组中相邻串的最大共同字串的变化也只限于新的插入位置附近很小的区域内,即首字相同的串。可以仅更新首字符相同串对应的 suftab 数组和 lcptab 数组中的元素,并遍历之,求得更新后的最长公共前缀子串。因此,算法的实现满足增量要求。

4 中文垃圾邮件过滤实验数据对比

我们将 SAC 应用于 PG 贝叶斯算法,并与切词方法抽取特征项后应用于朴素贝叶斯算法和 PG 贝叶斯算法进行了对比。

实验环境为:CPU 奔四 2.2G,内存 512M,硬盘 40G,操作系统为 Windows 2000 Server。实验方法采用 10 次交叉验证。

实验语料:

由于国内在中文垃圾邮件语料库建设上的滞后,尚未有公开的语料集,因此采用个人收集的中文垃圾邮件。为降低语料对个人邮箱的依赖性,语料分别来自不同的实验室收集的垃圾邮件,共计 2000 封(合法邮件与垃圾邮件比例为 1:1)。中文邮件的词语切分采用中国科学院计算技术研究所研制的汉语词法分析系统 ICTCLAS 软件包。

实验 1:由于朴素贝叶斯算法下邮件过滤的查准率和查全率与特征项集合的规模大小有关,实验 1 中列出了其在中文切词方式下不同特征项集合规模下的分类查准率和查全率。

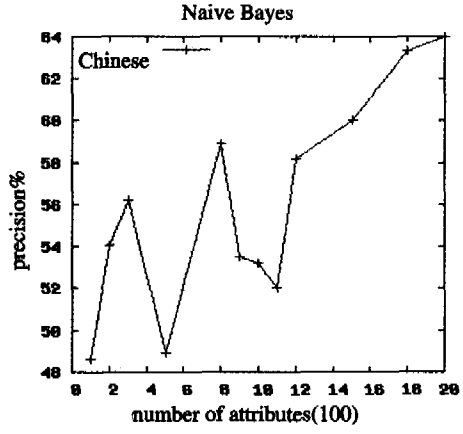

图 4 中文(切词)朴素贝叶斯分类查准率

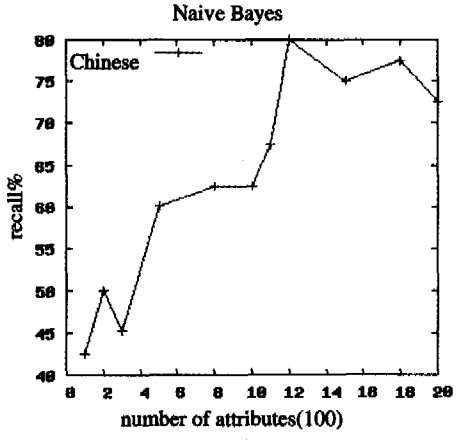

图 5 中文(切词)朴素贝叶斯分类查全率

从图 4、图 5 可以看出,当特征项集合的规模较小时,系统判断的"参照"点少而导致系统的查准率和查全率变化幅度大;当中文语料选取特征项集合的规模为 1200~1800 时,查准率和查全率都相对稳定;随着特征项的进一步增多,噪声数据也增多,查全率指标反而下降;此外,特征集合规模越大,系统的计算代价越高。因此,根据实验 1 的结果,选取朴素贝叶斯算法中分类指标高的点与 SAC 应用于 PG 贝叶斯算法进行对比。

实验 2:朴素贝叶斯算法和 PG 贝叶斯算法在中文切词下的指标和 SAC 特征抽取后应用于 PG 贝叶斯算法的指标对比。其中,朴素贝叶斯算法选取特征集合规模为 1500 和 1800。

可以看出,经 SAC 特征抽取后的中文邮件比用中文切词处理后的邮件应用分类算法效果有了显著的提升:系统的查全率和查准率都有很大提高;特别是系统的查准率,这一点在

(下转第 112 页)

向量为$\langle x_1, x_2, \cdots, x_n \rangle$，DLB模型计算特征向量分为两级，$L_1 = \langle x_1, x_2, \cdots, x_k \rangle$和$L_1 = \langle x_{n-k}, \cdots, x_n \rangle$，在运算规模上DLB分类模型要小于朴素贝叶斯分类模型，特别是当n很大时更为明显。表1显示了实验的结果。

表1 实验结果

	λ	R_{acc}(%)	R_{err}(%)	R_{miss}(%)
NB	1	96.8	0.44	1.1
DLB		96.5	0.39	0.8
Baseline (no filter)		53	0	50
NB	9	97.6	0.32	1.5
DLB		98.0	0.26	1.4
Baseline (no filter)		53	0	50
NB	999	7.9	0.25	2.2
DLB		98.4	0.17	2.0
Baseline (no filter)		53	0	50

从实验中我们发现DLB分类模型的学习时间要大于朴素贝叶斯分类模型所需要的时间，但是在大多数条件下，分类的准确率要优于朴素的贝叶斯分类模型，同时其错误率和漏报率要小于朴素贝叶斯分类模型。

总结 本文介绍了朴素的贝叶斯分类模型在邮件过滤中的应用，并且在此基础上提出改进的双级贝叶斯(DLB)分类模型，给出了改进的分类算法，并且进行仿真实验。通过实验结果可以看出新的模型在大部分性能指标上都优于一般的朴素贝叶斯分类模型。我们的下一步研究工作将进一步分析特征参数之间的相互信息对分类效果的影响，并考虑对邮件分类后进行回馈学习对模型分类性能的影响。

参考文献

1 Huang Cecil, Darwiche A. Inference in Belief Networks: A Procedural Guide. Int Journal of Approximate Reasoning, 1996, 15: 255～263

2 Friedman N, Geiger D, Goldszmidt M. Bayesian Network Classifiers. Machine Learning, 1997, 29(2-3): 131～163

3 Yerazunis W S. The Spam-Filtering Accuracy Plateau at 99.9% Accuracy and How to Get Past it. January 2004. Presented at the 2004 MIT Spam Conference

4 PaulGraham. com. a Plan for spam. www. paulgraham. com/spam. html

5 PaulGraham. com. Better Bayesian Filtering. www. paulgraham. com/better. html

6 Androutsopoulos I, Paliouras G, Karkaletsis V, et al. Learning to Filter Spam E-Mail: A Comparison of a Naive Bayesian and a Memory-Based Approach. In: Proc. of the Workshop on Machine Learning and Textual Information Access, 4th European Conf. Principles and Practice of Knowledge Discovery in Databases (PKDD 2000), Lyon, France, 2000. 1～13

7 Hastie T, Tibshirani R, Friedman J[美]著. 范明，柴玉梅等译. 统计学学习基础——数据挖掘、推理与预测. 北京：电子工业出版社，2004

8 石洪波，王志海，等. 一种限定性的双层贝叶斯分类模型. 软件学报，2004, 15(2)

(上接第109页)

垃圾邮件处理中尤为重要。分析其原因是：SAC根据数据本身"抱团"的性质来选取特征项，不但考虑短语出现的位置和顺序，而且特征项的长度仅仅由特征项本身决定。

指标 / 指标	查准率	查全率	分类时间(秒/每邮件)
SAC[PG贝叶斯分类算法]	98.9%	95.1%	0.523
切词[PG贝叶斯分类算法]	68.9%	64.6%	0.271
切词[朴素贝叶斯(1500)]	60.6%	75.4%	0.486
切词[朴素贝叶斯(1800)]	63.3%	77.5%	0.521

图6 切词和SAC下中文垃圾邮件过滤指标对比

因此，经SAC抽取后得到的特征项"分类质量"高，系统的输出指标有了明显的提高。

此外，使用SAC的另外一个优势在于系统不需要切词软件的支持和安装词典库。

应用SAC分类时间略长于其他方法。这一点可以通过增加"时间窗口"的方法加以解决。即对输入的语料增加时间参数的限制，使系统只处理一定时间范围内的语料。这样做与垃圾邮件在一定时间内具有重复性的特点是一致的。

结束语 本文针对中文垃圾邮件过滤指标低的问题，提出了一种新的基于后缀数组聚类的中文邮件特征项抽取方法SAC，并将其应用于PG贝叶斯算法，实现中文垃圾邮件的过滤。实验表明，系统的查准率和查全率得到显著提高。下一步，我们需要进一步缩短算法的分类时间，使得算法能够在大规模邮件过滤上获得更好的性能。

参考文献

1 Sahami M, Dumais S, Heckerman D, et al. A Bayesian Approach to Filtering Junk E-mail. In: AAAI Workshop on Learning for Text Categorization, Madis on, Wisconsin, 1998. 55～62

2 Graham P. Better Bayesian filtering. URL: http://paulgraham. com/better. html, 2003

3 Graham P. A Plan for Spam. URL. http://paulgraham. com/spam. html, 2002

4 Segal R, Crawford J, Kephart J, et al. SpamGuru: An Enterprise Anti-Spam Filtering System. In: Proceedings of First Conference on Email and Anti-Spam (CEAS), Mountain View, CA. 2004. URL: http://www. ceas. cc/papers-2004/126. pdf

5 李国栋，李卫. 基于文本分类技术的垃圾邮件识别系统. 微电子学与计算机，2004, 21(6): 143～146

6 刘新斌，李俊. 一种基于N-gram组合的中文垃圾邮件过滤方法. 微电子学与计算机，2004, 21(12): 85～91

7 Zamir O, Etzioni O. Grouper: A dynamic clustering interface to web search results. Eighth International World Wide Web Conference, TorontTo, 1999

8 Gusfield D. Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. furst edition. In: New York, USA: published by the press syndicate of the university of Cambridge, 1997. 90～91

9 Ukkonen E. On-line construction of suffix-trees. Algorithmica, 1995, 14(3): 249～260

10 Manber U, Myers G. Suffix arrays: A new method for on-line string searches. In: Proceedings of the First Annual ACM_ SIAM Symposium on Discrete Algorithms, 1990. 319～327

11 Abouelhoda M, Kurtz S, Ohlebusch E. Replacing Suffix Trees with Enhanced Suffix Arrays. Journal of Discrete Algorithms, 2004, 2(1): 53～86