计算机研究与发展
Journal of Computer Research and Development
ISSN 1000-1239/ CN 11-1777/ TP
43(10): 1681~ 1687, 2006

基于词频分类器集成的文本分类方法

姜 远 周志华
(南京大学软件新技术国家重点实验室 南京 210093)
(南京大学计算机科学与技术系 南京 210093)
(jiangyuan@nju.edu.cn)

A Text Classification Method Based on Term Frequency Classifier Ensemble

Jiang Yuan and Zhou Zhihua
(*National Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210093)
(*Department of Computer Science and Technology, Nanjing University, Nanjing* 210093)

Abstract In this paper, a method of text classification based on term frequency classifier ensemble is proposed. Term frequency classifier is a kind of simple classifier obtained after calculating terms' frequency of texts in the corpus. Though the generalization ability of term frequency classifier is not strong enough, it is a qualified base learner for ensemble because of its low computational cost, flexibility in updating with new samples and classes, and the feasibility of improving generalization with the help of ensemble paradigms. An improved AdaBoost algorithm is used to build the ensemble, which employs a scheme of compulsive weights updating to avoid early stop. Therefore it is more suitable for text classification. Experimental results on the corpus of Reuters-21578 show that the proposed method can achieve good performance in text classification tasks.

Key words text classification; machine learning; ensemble learning; term frequency classifier; AdaBoost

摘 要 提出了一种基于词频分类器集成的文本分类方法. 词频分类器是在对文本中的单词和它在每个文本中出现的频率进行统计后得到的简单分类器. 虽然词频分类器本身泛化能力不强,但它不仅计算代较小,而且在训练样本甚至类别增加时易于进行更新,而整个学习系统的泛化能力可以由集成学习机制来提高,因此,词频分类器很适合用做集成学习的基分类器. 在集成时,使用了改进的AdaBoost算法,加入了一种强制重新分布权的机制,避免算法过早停止,更加适合文本分类任务. 在标准文集Reuters-21578上的实验结果表明,该方法能取得很好的效果.

关键词 文本分类; 机器学习; 集成学习; 词频分类器; AdaBoost

中图法分类号 TP18

随着Internet技术的发展,越来越多的信息呈现于网页中,随之而来的问题是面临大量的信息时,如何快速而有效地对所需的信息进行检索. 现有的许多搜索引擎例如Google, Yahoo, WebCrawler, Lycos等在对信息进行检索中承担了重要的角色. 在网页文档中包含有大量的文本内容的信息,而对文本的检索[1]、归类[2]、选择[3]、过滤[4]等,通常是基于文本分类而进行的,这使得有效而准确的文本分类技术成为关键的所在.

文本分类的任务是将文集(corpus)中的文本分到预先定义的类别中. 通常的情况下,将一些已经具有类别标记的文本作为训练数据,学习系统在

收稿日期: 2006-04-29; 修回日期: 2006-05-29
基金项目: 国家自然科学基金项目(60505013); 江苏省自然科学基金创新人才基金项目(BK2005412)

学习之后能够将新的文本按其最大的相似度分到某个类中. 简单贝叶斯方法、决策树、K 近邻、支持向量机等机器学习方法都已被成功地应用于文本分类[5]. 也有学者尝试使用多种不同分类器的组合来进行文本分类[6-7]. 20 世纪 90 年代末, 集成学习(ensemble learning)技术开始进入该领域, 并成为该领域的一个研究热点, 例如, Weiss 等人[8]用决策树的集成进行文本分类, 并且成功地用于 email 的过滤. Schapire 等人[9]将决策树桩(decision stump)的集成用于文本分类系统 BoosTexter 中, 也取得了较好的效果.

本文提出了一种基于词频分类器集成的文本分类方法. 集成中使用的词频分类器不仅计算开销小, 而且在增加文本样本甚至类别时, 可以容易地进行更新. 在 Reuters-21578 标准文集上的实验结果表明, 该方法可以取得很好的文本分类性能.

1 集成学习

集成学习通过训练基学习器的多个版本来解决同一个问题. 由于集成通常能够得到比单个学习器更强的泛化能力, 因此, 对集成学习的研究被 Dietterich 认为是当前机器学习的四大研究方向之首[10]. 集成的构建通常包括两个步骤, 首先是利用基学习器训练出多个版本, 即得到多个个体学习器, 然后将这些个体学习器进行结合. 按照个体学习器的生成方式的不同, 可以将集成方法大致分为两类[11]: 一类以 AdaBoost[12]为代表, 在这一类方法中, 个体学习器是顺序生成的, 上一轮的学习器的结果将会影响到其后的学习器的生成, 属于这一类的还有 Arc-x4[13], MultiBoost[14]等集成方法. 另一类以 Bagging[15]为代表, 个体学习器的生成是并行的, 相互不受干扰, 属于这一类的集成方法还有 Wagging[16], p-Bagging[16], GASEN[17]等. 此外, 集成学习也被成功地应用到字符识别[18]、人脸识别[19]、图像分析[20]、医疗诊断[21]等应用领域.

本文工作主要涉及 AdaBoost 算法[12], 该算法将数据集$\{(x_1, y_1), \cdots, (x_m, y_m)\}$作为其训练数据, 其中, x_i 是示例空间X 中的示例, y_i 是x_i 的概念标记, 这里 Y 定义在$\{-1, +1\}$上. 在确定基学习器之后, AdaBoost 重复地调用基学习器, 在每一轮的调用中, AdaBoost 算法对训练集上的数据分布进行调整, 这可以通过维护一个权集合来实现. 在初始的状态下, 示例的权是相等的, 在以后的各轮中, 上一轮被错误分类的示例将被赋予更大的权, 以使得这些较难的示例被更多地关注. 具体的算法细节请参见文献[12].

2 词频分类器及其集成

2.1 文本分类的特殊性

文本信息的形式多样且信息量大, 如果以每个词为一个特征而言, 对大批量的文本进行分类实质是一个在高维空间中对高维特征向量进行分类的任务. 在面临开放应用的情况下, 随时会有可利用的新的样本加入, 这些新的样本甚至可能会属于新的类别, 这时, 要求文本分类系统能够充分利用这些新的样本而尽可能降低更新所带来的代价. 由于文本分类经常是在线进行的, 这就要求文本分类系统具有很好的实时性.

2.2 理想的基分类器所应具有的性质

针对文本分类的上述特点, 可以看出, 理想的文本分类学习算法应该具有分类精度高、计算代价小、计算速度快和易于进行数据更新的性质. 通常, 复杂的学习算法为了达到分类精度高的要求, 在计算代价和计算速度上需要很大的代价. 简单的学习算法相较而言可以降低计计算复杂度, 提高计算速度, 然而分类精度都较差. 而根据新数据进行更新则是大多数学习算法所面临的共同难题.

由于集成学习技术可以将弱分类器提升为泛化能力很强的强分类器, 在集成中使用的基分类器就不再需要是计算代价高、更新困难的复杂分类器, 而可以是更符合文本分类特殊要求的快速、高效且易于更新的基分类器, 而基分类器的泛化能力则不必很强.

2.3 词频分类器

我们可以根据单词的出现定义一个基分类器. 具体来说, 在所有文本组成的文集上可以得到一个所有出现的单词组成的词汇表 V. 每个文本 $\boldsymbol{d}_i$ 表示成形如$(w_{i1}, \cdots, w_{in})$的矢量, 其中 w_{ik} 表示 V 中的第 k 个单词 t_k 是否出现在文本 $\boldsymbol{d}_i$ 中, 如果出现, 则 $w_{ik}=1$, 否则 $w_{ik}=0$. 这样, 基分类器可以被定义为

$$h_{t_k}(\boldsymbol{d}_i)=\begin{cases}1, & \text{if } w_{ik}=1,\\ 0, & \text{if } w_{ik}=0.\end{cases} \tag{1}$$

本文称这种基分类器为词出现分类器(term occurrence classifier, TOC), 需要注意的是词出现分类器 TOC 只考虑了单词在文本中出现与否, 因此一个

单词在某文本中出现一次和出现多次其函数值都为1. 而在实际情况中, 单词在文本中的出现频率是传递了一定意义的. 一般而言, 在排除了干扰词表(stop list)中的词后, 某个单词在一个文本中出现的频率越高, 则对该文本的分类影响越大.

因此, 我们可以将单词和它出现的频率一起作为一个基分类器. 若文本 $\boldsymbol{d}_i$ 表示为矢量$(v_{i1}, \cdots, v_{in})$, v_{ik}表示 V 中的第 k 个单词 t_k 在文本$\boldsymbol{d}_i$ 中出现的次数, 此时, 基分类器被定义为

$$h_{t_k,f}(\boldsymbol{d}_i) = \begin{cases} 1, & \text{if } v_{ik} \geqslant f, \\ 0, & \text{if } v_{ik} < f. \end{cases} \qquad (2)$$

本文称这种基分类器为词频分类器(term frequency classifier, TFC), 这里 f 是单词 t_k 出现在文本中的频率. 对一个文集来说, 同一个词在不同的文本中有不同的词频, 这样, 根据每一个词及其每一种可能的词频, 都有一个对应的基分类器版本. 假设文集中有 M 个词, 则词出现分类器 TOC 共有 M 个可能的版本; 假设文集中第 k 个词的可能的词频的数目为 l_k, 则词频分类器 TFC 共有 $\sum_{k=1}^{M} l_k$ 个可能的版本. 显然 $\sum_{k=1}^{M} l_k \gg M$, 即可供集成选择的基分类器的版本变多了, 这为集成学习的处理提供了便利.

值得注意的是, 词频分类器 TFC 具有计算代价小、计算速度快的特点. 这是由于对于其他类型的分类器而言, 在形成用于学习的训练数据时, 需要对文集中的文档和词的信息进行统计, 然后才进入训练阶段. 而词频分类器在对词频和文档信息进行统计之后, 就已经形成了分类器. 显然, 词频分类器的计算开销远小于其他类型的分类器. 此外, 由词频分类器的定义可以看出, 如果训练集中的文档发生变化时, 例如有新文档加入训练集时, 仅需要对新加入的文档中的词例和词频进行考察: 对于词汇表中已经出现过的词例和词频, 更新相应类别的计数; 而将未在词汇表中出现的词例和词频, 加入词汇表中并记录相应类别的计数. 这样就完成了词频分类器的更新. 若采用其他类型的分类器作为集成的基分类器, 例如, 在 Weiss 等人[8]的方法中采用决策树来作为集成的基分类器, 则如果训练集发生变化, 就需要对训练集中所有的数据进行重新训练, 从而产生新的决策树. 这样, 即使训练集只是增加了或更改了很少的一部分文档, 训练新的决策树所带来的计算开销也和最初在原有训练集上产生决策树的计算开销基本相同的. 相比而言, 词频分类器可以只针对新的内容进行快速更新, 由于在真实世界的文本分类问题中, 训练样本甚至类别都可能在应用中不断增加, 因此, 作为集成中的基分类器, 词频分类器在文本分类应用中具有很大的优势.

另外, 词出现分类器的思想在以往的基于集成的文本分类方法中也有使用, 例如 Schapire 等人的 BoosTexter 系统[9]中, 基学习器就是以单词的出现与否来对当前文本赋予一实数值, 用于进行文本的判别. 本文的后续部分将把这一方法与词频分类器进行比较.

2.4　改进的 AdaBoost 算法

在标准的 AdaBoost 算法[12]中, 第 t 轮中弱分类器的训练误差 ε_t 如式(3)所示:

$$\varepsilon_t = Pr_{i\sim D_t}[h_t(x_i) \neq y_i], \qquad (3)$$

其中, D_t 表示第 t 轮中各训练样本的权所满足的分布; h_t 为基分类器的当前版本; x_i 和 y_i 分别为示例及其类别.

在两类问题中, AdaBoost 算法最后得到的分类器 H 的训练误差最多为

$$\prod_t [2\sqrt{\varepsilon_t(1-\varepsilon_t)}] = \prod_t [\sqrt{1-4\gamma_t^2}] \leqslant \exp(-2\sum_t \gamma_t^2). \qquad (4)$$

由于对样本进行随机猜测的错误概率为 0.5, 式(4)中 $\gamma_t = 0.5 - \varepsilon_t$, 表示弱分类器与随机猜测之间的差距, 当 ε_t 比进行随机猜测的误差小时, 即 $\gamma_t > 0$ 时, AdaBoost 算法最后得到的分类器 H 的误差可以降低到非常小的值[12]. 因此, 在标准的 AdaBoost 算法中, 要求每一轮中弱分类器的训练误差必须小于 0.5, 否则训练过程就将停止.

值得注意的是, 在进行文本分类时, 很少有仅凭少数几个词就能进行正确分类的情况出现, 在大多数情况下, 需要有较多的具有区别力的词集合在一起, 才能够进行正确的分类. 由于词频分类器 TFC 本身是一种很弱的分类器, 其本身的分类能力并不强, 因此, 正需要利用集成学习技术将很多这样的弱分类器集成起来达到较强的分类能力. 而在标准的 AdaBoost 算法中, 由于其要求每一轮选出的弱分类器的误差都要小于 0.5, 这就使得在选出很少几个词频分类器之后, 学习过程就会停止, 这样, 最后得到的集成只包含很少的词频分类器, 其分类性能将受到很大的限制. 为了使得集成能够利用更多的词频分类器, 本文对标准的 AdaBoost 进行了改进. 具体来说, 本文设计了一种强制重新分布权的机制,

在当前的弱分类器误差未能满足小于 0.5 的情况下,算法直接对当前的样本权分布重新调整,调整的方法可以有多种,例如重新给样本随机分配权,再计算误差,符合条件的话算法继续进行,否则再次尝试权的重新分配,直到条件满足. 或者,在上轮的样本分布的基础上进行适当的调整. 本文采用的是用初始权来替换当前的样本权的方式,使算法继续进行下去.

在算法的初始状态下,对由文本 $\boldsymbol{x}_i$ 和它的类标记 y_i 组成的训练集 $\{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_m, y_m)\}$ 中的每个正例赋权 $1/(2k)$,每个反例赋权 $1/(2l)$,其中 k, l 分别是正例和反例的个数. 这样赋值的目的是在正例和反例的数量相差悬殊时,数量少的一方所传递的信息不至于被数量多的一方所淹没. 之后,根据字典 V 中的每一个词 j 都可以构建出一个弱分类器 h_j. 学习算法在循环部分的每一轮里,都按最小化假设误差的原则来挑选出一个弱分类器 h_t,在经过若干轮之后,得到相应的一组弱分类器. 算法最后对这一组弱分类器进行结合,最终得到一个精度较高的分类器. 算法流程如下所示,其中 h_j 既可以由词频分类器 TFC 作为基分类器产生,也可以由词出现分类器 TOC 作为基分类器产生.

算法 **1**. 改进的 AdaBoost 算法:

Given:

training set $\{(\boldsymbol{x}_1, y_1), \cdots, (\boldsymbol{x}_m, y_m)\}$, where $\boldsymbol{x}_i \in X$, $y_i \in Y$.

Initialization:

If $\boldsymbol{x}_i$ is positive, then set its initial weight $w_1(\boldsymbol{x}_i) = 1/(2k)$, k is the number of postive examples; if $\boldsymbol{x}_i$ is negtive, then set its initial weight $w_1(\boldsymbol{x}_i) = 1/(2l)$, l is the number of negtive examples.

For $t = 1, \cdots, T$:

① For each $\boldsymbol{x}_i$, normalize $w_t(\boldsymbol{x}_i)$.

② For each version of base classifier, i.e. h_j, calculate its error:

$$\varepsilon_j = \sum_i w_i \mid h_j(\boldsymbol{x}_i) - y_i \mid .$$

③ Select minimum error of this round ε_j as ε, h_j as h_t.

④ Calculate $\beta_t = \dfrac{\varepsilon_t}{1-\varepsilon_t}$.

⑤ If $\varepsilon_t < 0.5$, then update $w_i(t+1) = w_i(t)\beta_t^{1-|h_t(\boldsymbol{x}_i)-y_i|}$, else assign the distribution of example weights according to the initial distribution.

Output the final classifier:

$$H(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{t=1}^{T}\alpha_t h_t(\boldsymbol{x})\right), \text{ where } \alpha_t = \log\left(\frac{1}{\beta_t}\right).$$

3 实 验

本文在文本分类领域的标准文集 Reuters-21578[22] 上进行了实验. 实验采用的是“ApteMod”版本的 Reuters-21578 文集,即从原始的文集中滤去没有标记的文档,并且只选择那些至少在训练集和测试集中各出现一个文档的那些类别来组成文集.

本文在 9 个类别的数据上进行了实验,使用的实验数据的情况如表 1 所示:

Table 1 Experimental Data

表 **1** 实验数据的情况

Class Name	Number of Training Examples	Number of Testing Examples
Earn	2877	1087
acq	1650	719
money-fx	538	179
grain	433	149
crude	389	189
trade	368	118
interest	347	131
wheat	212	71
ship	197	89

本文参照 Schapire 等人[9] 的方式使用实验数据,即逐渐增加数据集的类别数并比较不同类别数下的实验结果. 具体来说,本文分别以表 1 中的前 3 个类为正例组织数据,对这 3 个类中的每个类依次分别加入其他 2~8 个类的样本作为反例,也就是以每个类为目标类可以形成 7 个不同的数据子集,对于每个类别来说,每个文档或者被判为属于该类,或不属于该类. 这样,本文就得到了 21 组实验数据集. 在每组数据集上,本文分别对词频分类器与词出现分类器 TOC、改进 AdaBoost 算法和标准 AdaBoost 算法的 4 种组合进行了测试,此外,本文还对经典的文本分类方法 TF-IDF[1] 进行了测试.

本文使用的评测指标为查准率 *Precision*、查全率 *Recall* 以及 $F1$[23],结果分别如表 2 至表 4 以及图 1 所示. 需要注意的是,*Precision*, *Recall*, $F1$ 这 3 个指标的值都是越高越好.

Table 2　Comparison of *Precision* and *Recall* on Class "earn"

表 2　earn 类上 *Precison* 和 *Recall* 值的比较

No. Classes	TF-IDF		TOC+ AdaBoost		TOC+ Improved AdaBoost		TFC+ AdaBoost		TFC+ Improved AdaBoost	
	Precision	*Recall*	*Precision*	*Recall*	*Precision*	*Recall*	*Precision*	*Recall*	*Precision*	*Recall*
3	0.8939	0.8939	0.9469	0.9199	0.9483	0.9457	0.9617	0.9264	0.9644	0.9245
4	0.8801	0.8801	0.9380	0.9199	0.9378	0.9438	0.9484	0.9310	0.9784	0.9172
5	0.8755	0.8755	0.9302	0.9199	0.9365	0.9365	0.9440	0.9310	0.9847	0.8914
6	0.8755	0.8755	0.9293	0.9199	0.9373	0.9365	0.9396	0.9310	0.9790	0.9015
7	0.8596	0.8596	0.9285	0.9199	0.9393	0.9402	0.9450	0.9181	0.9868	0.8960
8	0.8596	0.8596	0.9285	0.9199	0.9429	0.9429	0.9450	0.9181	0.9818	0.8942
9	0.8596	0.8596	0.9267	0.9199	0.9457	0.9457	0.9387	0.9310	0.9532	0.9199
Avg	0.8720	0.8720	0.9325	0.9199	0.9411	0.9416	0.9460	0.9266	0.9754	0.9063

Table 3　Comparison of *Precision* and *Recall* on Class "acq"

表 3　acq 类上 *Precison* 和 *Recall* 值的比较

No. Classes	TF-IDF		TOC+ AdaBoost		TOC+ Improved AdaBoost		TFC+ AdaBoost		TFC+ Improved AdaBoost	
	Precision	*Recall*	*Precision*	*Recall*	*Precision*	*Recall*	*Precision*	*Recall*	*Precision*	*Recall*
3	0.6342	0.6342	0.5964	0.6022	0.7218	0.7218	0.6699	0.6717	0.7927	0.7927
4	0.5674	0.5674	0.5924	0.5924	0.7148	0.7148	0.6629	0.6648	0.7597	0.7607
5	0.5563	0.5563	0.5884	0.5785	0.7009	0.7009	0.6609	0.6481	0.7184	0.7134
6	0.5479	0.5479	0.5877	0.5591	0.7093	0.7093	0.6313	0.6383	0.7218	0.7218
7	0.5424	0.5424	0.5489	0.5618	0.7176	0.7176	0.6288	0.6314	0.7218	0.7218
8	0.5340	0.5340	0.5489	0.5618	0.7315	0.7315	0.6288	0.6314	0.7426	0.7426
9	0.5299	0.5299	0.5474	0.5618	0.7426	0.7426	0.6253	0.6314	0.7496	0.7496
Avg	0.5588	0.5588	0.5728	0.5739	0.7197	0.7197	0.6439	0.6453	0.7438	0.7432

Table 4　Comparison of *Precision* and *Recall* on Class "money-fx"

表 4　money-fx 类上 *Precison* 和 *Recall* 值的比较

No. Classes	TF-IDF		TOC+ AdaBoost		TOC+ Improved AdaBoost		TFC+ AdaBoost		TFC+ Improved AdaBoost	
	Precision	*Recall*	*Precision*	*Recall*	*Precision*	*Recall*	*Precision*	*Recall*	*Precision*	*Recall*
3	0.6871	0.6871	0.6440	0.4245	0.6759	0.6759	0.8275	0.4022	0.7289	0.6759
4	0.5865	0.5865	0.6333	0.4245	0.6648	0.6648	0.8275	0.4022	0.7687	0.6312
5	0.4916	0.4916	0.6229	0.4245	0.6592	0.6592	0.8181	0.4022	0.76	0.6368
6	0.4636	0.4636	0.5937	0.4245	0.6312	0.6312	0.4022	0.5353	0.7552	0.6033
7	0.4413	0.4413	0.6153	0.4022	0.553	0.553	0.6153	0.4022	0.5842	0.5810
8	0.2737	0.2737	0.6153	0.4022	0.581	0.581	0.6153	0.4022	0.5639	0.5418
9	0.2737	0.2737	0.6101	0.4022	0.5865	0.5865	0.6101	0.4022	0.5154	0.5586
Avg	0.4596	0.4596	0.6192	0.4149	0.6216	0.6216	0.6737	0.4212	0.6680	0.6040

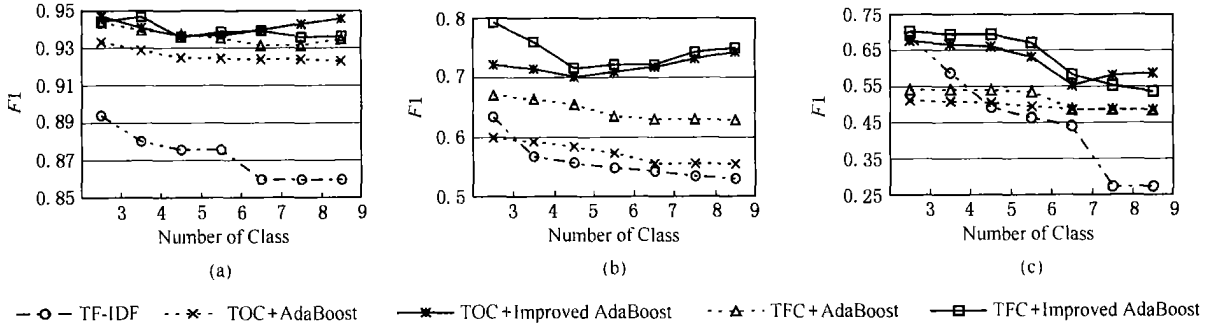


Fig. 1　Comparison of $F1$ values. (a) $F1$ on class "earn"; (b) $F1$ on class "acq"; and (c) $F1$ on class "money-fx".

图 1　$F1$ 值的比较　(a) "earn" 类上 $F1$ 值;(b) "acq" 类上 $F1$ 值;(c) "money-fx" 类上 $F1$ 值

从实验结果可以看出, 对于 acq 和 money-fx 这两个类, 本文方法(即词频分类器 TFC+ 改进 AdaBoost)明显好于其他方法; 而对 earn 这个类, 由于正例数目较多, "词出现分类器 TOC+ 改进 AdaBoost"这一方法的性能有所提高, 与本文方法性能相当, 但本文方法仍然明显好于其他方法. 总的来看, 在使用同样的集成算法时(无论是标准 AdaBoost 还是改进 AdaBoost), 使用词频分类器 TFC 的结果要优于使用词出现分类器 TOC 的结果; 而在使用同一种基分类器时(无论是词频分类器 TFC 还是词出现分类器 TOC), 使用改进 AdaBoost 的结果要优于使用标准 AdaBoost 的结果. 这充分说明词频分类器 TFC 和改进 AdaBoost 的结合是一种较好的选择.

4 结束语

本文提出了一种基于词频分类器集成的文本分类方法. 针对文本分类应用的特殊性而设计的词频基分类器 TFC 与常用的分类器相比, 不仅训练时间少、计算代价小, 而且在有新的训练样本甚至类别加入时, 只需简单地加入新增词频就完成了更新, 不需要对整个分类器重新训练, 是一种适于文本分类的基分类器. 在集成时, 为了避免因为当前个体学习器未能符合假设误差的要求而过早地终止循环, 本文设计了一种强制重新分布样本权的机制, 对 AdaBoost 算法进行了改进. 在标准文集 Reuters-21578 上的实验结果表明, 词频分类器 TFC 与改进的 AdaBoost 算法相结合可以取得很好的文本分类效果. 本文方法在集成的每一轮循环中, 对基分类器版本的选择是按照最小化训练误差的原则进行的. 在进一步的工作中, 我们准备尝试使用不同的代价函数来进行基分类器版本的选择, 以期进一步提高本文方法进行文本分类的性能.

参 考 文 献

[1] G Salton. Development in automatic text retrieval [J]. Science, 1991, 253(5023): 974-980

[2] L L Diao, K Y Hu, Y C Lu, *et al*. Improved stumps combined by boosting for text categorization [J]. Journal of Software, 2002, 13(8): 1361-1367

[3] S Wermter, G Arevian, C Panchev. Recurrent neural network learning for text routing [C]. The Int'l Conf on Artificial Neural Networks, Edinburgh, UK, 1999

[4] Ma Liang, Chen Qunxiu, Cai Lianhong. An improved model for text information filtering [J]. Journal of Computer Research and Development, 2005, 42(1): 79-84 (in Chinese)
(马亮, 陈群秀, 蔡莲红. 一种改进的自适应文本信息过滤模型[J]. 计算机研究与发展, 2005, 42(1): 79-84)

[5] F Sebastiani. Machine learning in automated text categorization [J]. ACM Computing Surveys, 2002, 34(1): 1-47

[6] Y H Li, A K Jain. Classification of text documents [J]. The Computer Journal, 1998, 41(8): 537-546

[7] Tang Chunsheng, Jin Yihui. A multiple classifiers integration method based on full information matrix [J]. Journal of Software, 2003, 14(6): 1103-1109 (in Chinese)
(唐春生, 金以慧. 基于权信息矩阵的多分类器集成方法[J]. 软件学报, 2003, 14(6): 1103-1109)

[8] S M Weiss, C Apte, F J Damerau, *et al*. Maximizing text-mining performance [J]. IEEE Intelligent Systems, 1999, 14(4): 63-69

[9] R E Schapire, Y Singer. Boostexter: A boosting-based system for text categorization [J]. Machine Learning, 2000, 39(2-3): 135-168

[10] T G Dietterich. Machine learning research: Four current directions [J]. AI Magazine, 1997, 18(4): 97-136

[11] Z-H Zhou, W Tang. Selective ensemble of decision trees [G]. In: Lecture Notes in Artificial Intelligence 2639. Berlin: Springer, 2003, 476-483

[12] Y Freund, R E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting [J]. Journal of Computer and System Sciences, 1997, 55(1): 119-139

[13] L Breiman. Arcing classifiers [J]. Annals of Statistics, 1998, 26(3): 801-849

[14] G I Webb. MultiBoosting: A technique for combining boosting and wagging [J]. Machine Learning, 2000, 40(2): 159-196

[15] L Breiman. Bagging predictions [J]. Machine Learning, 1996, 24(2): 123-140

[16] E Bauer, R Kohavi. An empirical comparison of voting classification algorithms: bagging, boosting, and variants [J]. Machine Learning, 1999, 36(1-2): 105-139

[17] Z-H Zhou, J Wu, W Tang. Ensembling neural networks: Many could be better than all [J]. Artificial Intelligence, 2002, 137(1-2): 239-263

[18] J Mao. A case study on bagging, boosting and basic ensembles of neural networks for OCR [C]. The Int'l Joint Conf on Neural Networks, Anchorage, AL, 1998

[19] F J Huang, Z-H Zhou, H J Zhang, *et al*. Pose invariant face recognition [C]. The 4th IEEE Int'l Conf on Automatic Face and Gesture Recognition, Grenoble, France, 2000

[20] K J Cherkauer. Human expert level performance on a scientific image analysis task by a system using combined artificial neural networks [C]. The AAAI-96 Workshop on Integrating Multiple Models for Improving and Scaling Machine Learning Algorithms, Portland, OR, 1996

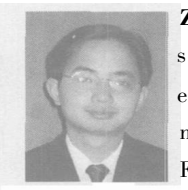
[21] Z-H Zhou, Y Jiang, Y B Yang, *et al*. Lung cancer cell identification based on artificial neural network ensembles[J]. Artificial Intelligence in Medicine, 2002, 24(1): 25-36

[22] D D Lewis. Reuters-21578 text categorization collection. http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html, 1999-02-16

[23] C J van Rijsbergen. Information Retrieval[M]. 2 edition. London: Butterworths, 1979

Jiang Yuan, born in 1976. Received her Ph D degree in computer science from Nanjing University in 2004. She is an associate professor of the Computer Science & Technology Department of Nanjing University. Her main research interests include machine learning, information retrieval and data mining.

姜远, 1976 年生, 博士, 副教授, 主要研究方向为机器学习、信息检索、数据挖掘等.



Zhou Zhihua, born in 1973. Ph D, professor and Ph D supervisor of the Computer Science & Technology Department of Nanjing University. Senior member of China Computer Federation. His current research interests include artificial intelligence, machine learning, data mining, information retrieval, pattern recognition, neural computing and evolutionary computing.

周志华, 1973 年生, 博士, 教授, 博士生导师, 中国计算机学会高级会员, 主要研究方向为人工智能、机器学习、数据挖掘、信息检索、模式识别、神经计算、演化计算等.

Research background

Text classification aims to assign documents with predefined categories automatically. Research on text classification has been flourishing due to the incredible increase of online documents in recent years. A number of machine learning approaches, such as Bayesian classifiers, neural networks, decision trees, support vector machines, etc., have been applied to text classification. In this paper, a novel method is proposed where text classification is achieved via ensemble of term frequecy classifiers. Experimental results show that the flexibility of term frequency classifier and the generalization ability of ensemble learning have both been utilized in the proposed method to obtain good performance in text classification tasks. Our work is supported by the National Natural Science Foundation of China (60505013) and the Jiangsu Science Foundation (BK2005412).