SAT-FOIL+:基于句子级关联的文本分类*)

冯玉才 李 曲 何 玉 冯剑琳

(华中科技大学计算机科学与技术学院 武汉430074)

摘 要 以往基于词语关联的方法在挖掘频繁项集和关联规则时,都是将整个文本看作一个事务来处理的,然而文本的基本语义单元实际上是句子。那些同时出现在一个句子里的一组单词比仅仅是同时出现在同一篇文档中的一组单词有更强的语义上的联系。基于以上的考虑,我们把一篇文档里的一个句子作为一个单独的事务,从而提出了一种基于句子级关联的分类方法SAT-FOIL。并在本文中提出新的得分模型来获得改进的新算法SAT-FOIL+。通过在标准的文本集Reuters上的大量实验,不仅证明新模型的优越性,而且证明了SAT-FOIL+分类效果同其他几种分类方法是可比的,并且要远远好于以往的基于文档级关联的分类方法。另外,挖掘出来的分类规则还具有易读性,并且易修改。

关键词 文本分类,句子级别,关联规则,频繁项目集

SAT-FOIL+: Sentence-Level Association Based Text Classification

FENG Yu-Cai LI Qu HE Yu FENG Jian-Lin

(Department of Computer Science, Huazhong University of Science and Technology, Wuhan 430074)

Abstract While previous association based methods mainly mined frequently co-occurring words (frequent itemsets) at the document-level, the basic semantic unit in a document is actually a sentence. Words within the same sentence are typically more semantically related than words that just appear in the same document. Our proposed SAT-FOIL views a sentence rather than a document as a transaction. In this paper we proposed new score models to get the improved algorithm SAT-FOIL+. The effectiveness of our proposed SAT-FOIL+ method has been demonstrated not only better than our former algorithm SAT-FOIL but also comparable to well-known alternatives and much better than previous document-level association based methods by extensive experimental studies using popular benchmark text collections Reuters. In addition, SAT-FOIL+ has inherent readability and refinability of acquired classification rules.

Keywords Text classification, Sentence-level, Association rules, Frequent itemsets

1 引言

随着电子化文本的持续增长,文本分类(Text Classification 或 Text Categorization)在我们的日常生活中变得越来越重要,已被广泛应用于文本检索,文本组织,邮件过滤,Web网页分层等领域。目前多种分类方法已经陆续被提出来,如贝叶斯网络,决策树,神经网络,支持向量机以及基于关联规则的分类方法等[1~8]。

从关联规则的角度来看,用户在进行手工分类时使用的一条形为"c ⇒ WHEAT"的分类规则实际上就是一条关联规则,例如:wheat AND farm⇒WHEAT。目前所有的基于关联规则的分类方法基于如下思想[5,7,15]:文档中的单词被看作是项目(item),例如:"wheat" 和"farm"是两个不同的项目;每一篇文档被看作是一个事务[3],即项目的集合。在不同文档中频繁出现的项目集(即:文档级的单词共同出现,简称共现)被用于捕捉文本的语义和产生分类的IF-THEN规则。然而,文档中的基本语义单元其实是句子。在同一个句子中共现的单词通常以这样或那样的方式相互关联,与跨越文档中多个句子的同一组单词相比,它通常更有意义。例如:一篇文档包含1000个句子,单词wheat出现在该文档的第一个自然句子中,同时出现在第一句中的单词farm将比出现在第518个句子中的farm有意义得多。

基于上述观察,我们在文[18]中提出了一种新的基于句子级关键词关联的文本分类方法:SAT-FOIL,它将一个自然语句看作是一个关联事务,而一篇文档则看作是它所有自然句子的集合。在本文中我们提出SAT-FOIL的改进算法SAT-FOIL+——一种新的分类得分模型,通过在标准的文本集Reuters上的大量实验,证明了新的得分模型有效地改善了分类结果,而且同其他几种分类方法是可比的,并且要远远好于以往的基于文档级关联的分类方法。

本文第2节介绍SAT-FoIL+的基本定义和框架。第3节详细介绍SAT-FOIL+。第4节提出新旧得分模型。第5节使用公共数据集Reuters上的实验说明SAT-FOIL+的有效性。第6节介绍了主要的相关工作。最后总结了全文。

2 基本定义和框架

令 $W=\{w_1,w_2,\cdots,w_M\}$ 为单词的集合,单词 w_i 称为项目。W 中若干项目的集合称为项目集,包括仅仅只有单个项目的项目集的情况。我们把项目集中单词的数目称为项目集的长度,称长度为 k 的项目集为 k-项目集。在本文中将只考虑英文文档,但SAT-FOIL+的基本思想也能被应用到其它语言,如中文。一个事务被定义为以".","!","?"和";"等句子结

*)本文受国家自然科学基金(编号60303030)和重庆自然科学基金(编号8721)支持。冯玉才 教授,主要研究领域为数据库与信息系统;李 曲 博士生,主要研究方向为文本挖掘;何 玉 硕士生,主要研究方向为文本挖掘;冯剑琳 博士,主要研究领域为联机事务处理与文本挖掘。

束符结束的一条自然语句，因而，一篇文档是一组事务的集合。每一个事务有一个唯一的标识符，称为 TID。一个 TID 由(DocID,SenID)对组成，其中 DocID 表示该事务所属于的文档，SenID 表示事务对应的语句在文档中的序号。

如果项目集 I 是事务 T 的子集，我们说 T 包含 I。如果文档 D 中 S_d 个事务包含 I，那么项目集 I 在文档 D 中的文档支持度(document Support)为 S_d。

定义1(文档频繁项目集，DFI) 项目集被认为是文档 D 的 DFI(文档频繁项目集)，是指其文档支持度大于用户指定的最小支持度(即文档最小支持度)。

当文档 D 中至少有一个事务包含 I 时，说文档 D 包含项目集 I。这里 DFI 用于表示语句级的频繁单词共现，并且被用于表示整个文档主题的一个侧面。直觉上，蕴涵文档主题的单词组(项目集)应该出现在构成文档的最小比例的语句中，为了充分地捕捉这种直觉，文档的最小支持度的设置应该确保一个 DFI 至少出现在2个句子中。因此，文档最小支持度2被称为自然文档最小支持度(natural document minsup)。

给定一篇文档 D 和用户指定的文档最小支持度 d_s，假定文档 D 经过了去停词、提取词干等预处理。任何在同一语句中出现了若干次的单词仅仅被计作一次，并被编码为一个整数。每一个事务中的项目按照字典序排序，项目集中的项目也是如此。我们采用 Borgelt 实现的 Apriori 算法[12]挖掘频繁项目集。Apriori 算法基于如下直觉[3,13]：在同一文档中，文档频繁项目集的任何子集一定也是文档频繁的。

3 构建分类器

3.1 频繁项目集

如果项目集 I 在类 C 的 s_c 个文档中是文档频繁的，项目集 I 在类 C 中的支持度为 s_c。当项目集 I 在文档 D 中是文档频繁的，称 D 为 I 所覆盖，同时 D 也被称为 I 的支持文档(supporting document)。

定义2 类 C 的类频繁项目集(CFI)是指类 C 的项目集，它在类 C 中的类支持度大于用户指定的最小支持度(称为类最小支持度)。自然地，类最小支持度的设置应该确保 CFI 至少在2个文档中是文档频繁的。这是因为 CFI 捕捉了同一个类不同文档之间的“公共文档主题”。然而，我们可能没有足够多的训练文档，因此，单个文档也可以表示类主题的一个方面。这意味着类支持度最小为1。

给定 M 个预定义类$\{C_1, C_2, \cdots, C_m\}$的集合，假定每一个类 C_i 包含 K_i 个训练文档。类 $C_i(1\leqslant i\leqslant M)$的关联规则是形为 $I\Rightarrow C_i$ 的蕴涵式，其中 I 是 C_i 的 CFI。I 可以是几个类的 CFI，也可以是某个类的一些文档的 DFI 却不是该类的 CFI，因此，对整个训练文档集而言，我们需要确定其预计分类的置信度。

定义3 类频繁项目集 I 在类 C_i 中的置信度，表示为 $Conf(I\Rightarrow C_i)$，定义为 S_i 与 S_{tot}的比值，其中 S_i 为 I 在 C_i 中的支持度，S_{tot}为整个训练文档集中，I 所覆盖的不同的支持文档的总数。即 $Conf(I\Rightarrow C_i)=S_i/S_{tot}$。

对类 C_i 而言，I 所覆盖的文档被称为 I 关于类 C_i 的正例，I 在其它类所覆盖的其它文档被称为负例。因此，S_i 实际上是 I 的正例数，而 S_{tot}是去重后，正例数与负例数的总和。

3.2 构造类前缀树

给定类 C 的 K 个训练文档和用户指定的类最小支持度 c_s，现在说明如何使用类前缀树来收集类(频繁)项目集。在类前缀树中，每一条边标记为一个项目，每一个节点包含了从根节点到该节点的路径上的边所对应的项目集合。简单起见，我们将在下文中交替使用节点和(它对应的)项目集。例如，图1表示了一棵类前缀树。

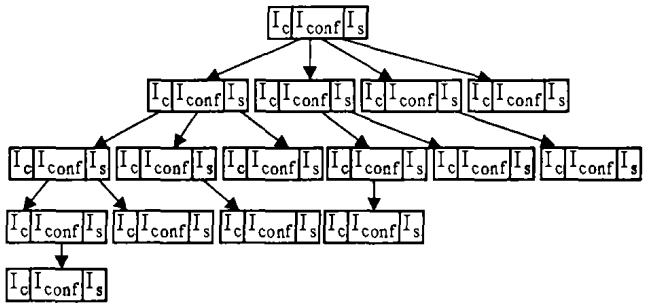

图1 包含4个项目1,2,3,4的前缀树

如图1所示，每一个节点包含3个计数器：I_c，I_{conf}和 I_s。我们使用 I_c 和 I_{conf}分别表示节点所对应的项目集 I 的类支持度和置信度。计数器 I_s 用于分类阶段，统计项目集 I 在新文档 D_n 中的文档支持度。此外，我们保存了与每一个节点相关联的文档的 *DocID* 链，*DocID* 链中记录了它的每一个支持文档的 *DocID*。这个 *DocID* 链也被称为节点的文档链。构建类前缀树的过程如下：

首先按照一篇训练文档中 DFI 的数量对类 C 中的所有 K 篇文档进行降序排序。然后选择含有 DFI 数目最多的文档 D_1，构造一棵前缀树保存所有的 DFI。每一个节点上的 I_c 计数器被初始化为1，同时该节点的 *DocID* 链中包含了一个文档号1，I_s 计数器被初始化为0。处理了 D_1以后，选择第二大的训练文档 D_2，在当前类前缀树上查找 D_2中的每一个 DFI，如果找到了该 DFI，就将对应的 I_c 加1，它表示该 DFI 为同一个类中的另一个训练文档所包含，同时将 D_2的文档号2加入到节点的 *DocID* 链中；否则为 DFI 创建一个新的节点并将它插入到前缀树中，其 I_c和 I_s 被分别初始化为1和0，同时该节点的 *DocID* 链中包含了一个文档号2。新的树被用来处理 D_2中的下一个节点(DFI)。当 K 个训练文档处理完毕，这个过程即结束。

3.3 构建分类器

给定 M 个预定义的类，可以使用以上的过程逐个构造 M 棵类前缀树。同时，构造一棵全局前缀树以收集在所有训练文档集中发现的每一个类项目集。这棵全局树用于计算 S_{tot}，它表示在整个训练文档集中类项目集覆盖的不同支持文档的总数。构造 M 棵类前缀树和全局树的步骤如下：

1)首先为第一个类构造一棵类前缀树，接着拷贝这棵类前缀树作为初始的全局类前缀树。然后，从第一棵类前缀树的每一个节点中移除 *docList* 以节约空间。

2)为下一个类构造一棵新的类前缀树，然后拷贝这棵树上的每一个节点，并在当前全局前缀树上查找节点拷贝 *c-node*。如果 *c-node* 匹配了节点 *g-node*，那么将 *c-node* 的文档链合并到 *g-node* 的文档链中，重复的文档被去除；否则，复制以 *c-node* 为根节点的子树，然后将该子树拷贝插入到全局类前缀树中的相应位置，以保证树上每个节点的子节点按字典序排列。当类前缀树的所有节点拷贝都处理完后，我们移除类前缀树上的 *docList* 链以节约空间。

3)对余下的($M-2$)个类，我们迭代执行步骤2。

一旦全部 M 棵类前缀树及全局树构造完毕，就使用定义3计算每一棵类前缀树节点的置信度(I_{conf})。此外，将销毁全

局前缀树以节约空间。

3.4 修剪类前缀树

给定一个预定义的类C,可以使用3.3节中构造的类前缀树作为最终的类分类器。把存储在类前缀树中的所有的非空项目集的数目称做这棵树(或分类器)的大小。一个理想的分类器的大小应该满足以下条件:一要足够大,以避免丢失一些重要的规则,二要尽量地小以避免过多的噪音和干扰。一个简单的解决办法就是设定一个分类器的最小置信度,如果一条规则的置信度小于这个值,则将它从前缀树中移去,但其难点就在于怎样设置一个合适的最小置信度,往往只能凭猜测。

一种较好的选择就是利用文档覆盖来修剪类前缀树,这种方法具有自我调节性,文档覆盖的基本思想来源于FOIL(First Order Inductive Learner)[9]。FOIL使用划分-覆盖策略来找到一些Horn子句,这些Horn子句覆盖大多数的(而不是全部的)正例,同时也覆盖少数(而不是没有)负例。我们这里采用FOIL的基本思想,提出了一种文档覆盖的修剪算法—DoCover。

DoCover算法的详细描述如图2,它通过函数FoilGen搜索到当前最好的项目集(第3行),接着移去被这个项目集覆盖的所有正例(第4,5行),然后再搜索当前最好的项目集,依次循环,直到这个类当中所有的正例都被覆盖了(第2行)。

我们用FoilGen来找到当前最好的项目集,它从一个种子项目集开始递归地扩展,每次扩展一项,直到不能再扩展下去或者是它不再覆盖任何的负例。在递归的最开始(DoCover算法的第3行),种子项目集被初始化为空,这样这个种子项目集就能覆盖所有的正例和负例文档了。我们借用FOIL gain来决定下一步应该扩展哪一个孩子节点。

Algorithm 1 DoCover
Input: The category prefix-tree of C (accessed by cRoot) and the global category prefix-tree (accessed by gRoot).
Globals:
numDocs: The total number of positive documents in the category which are still not covered;
cur: The current best itemset;
Output: The pruned category prefix Tree of C
1: *numDocs*=sizeof(*cRoot*->docList);
2: while(*numDocs*>0)
3: *cur*=*FoilGen*(*cRoot*, *gRoot*);
4: remove the positive documents already covered by *cur*;
5: update *numDocs*;
6: end while

图2 文档覆盖修剪算法

Procedure 1 FoilGen(cSeedNode, gSeedNode)
Globals:
maxGain: The current biggest gain;
cNewNode: the host node in the category prefix-tree which has the *maxGain*;
gNewNode: the host node in the global category prefix-tree which has the *maxGain*;
Output: The host node of the current best itemset.
1: Seach for unmarked *cNewNode* among all the children of *cSeedNode* referring to *gNewNode* for computation of FOIL gain;
2:*numPos*=sizeof(*cNewNode*->docLi)
3:*numTot*=sizeof(*gNewNode*->docList);
4:*numNeg*=*numTot*−*numPos*;
5:If (*numNeg*==0) or (*cNewNode* has not any child)
6: *cNewNode* is marked as the current best;
7: return *cNewNode*;
8: end if
9: return FoilGen(*cNewNode*, *gNewNode*);

图3 FoilGen过程

假设有$|P|$个正例和$|N|$个负例被种子项目集覆盖,现在在种子项目集上扩展一项,得到一个新的项目集*new*,假设*new*覆盖$|P'|$个正例和$|N'|$个负例,项目集*new*的*Foil gain*定义如下:

$$gain(new)=|P'|(\log\frac{|P'|}{|P'|+|N'|}-\log\frac{|P|}{|P|+|N|})$$

假设种子项目集在类前缀树和全局树上对应的节点分别为*cSeedNode*和*gSeedNode*。*cSeedNode*的文档链*docList*中存放的是所有被这个种子项目集覆盖的正例,I_c计数器的值实际上就是这些正例文档的数目$|P|$。同样地,*gSeedNode*的文档链*docList*中存放的是所有被这个种子项目集覆盖的正例和负例,这些正例和负例的总数目等于$(|P|+|N|)$。

直觉上,从种子项目集开始每次扩展一项,就相当于在*cSeedNode*的孩子节点中寻找最好的项目,因此*FoilGen*的思想是非常直观的。*FoilGen*的具体描述如图3。首先在类前缀树和全局树上查找具有最大*FOIL gain*的节点,假设节点为*cSeedNode*和*gSeedNode*(第1行)。然后我们计算被*cSeedNode*对应的项目集覆盖的正例数(*numPos*,第2行)以及正例文档和负例文档的总数(*numTot*,第3行)。用*numTot*减去*numPos*得到负例文档的数目(*numNeg*,第4行)。如果没有负例文档被覆盖或项目集没有新的项目用于扩展,我们认为*cNewNode*已经代表了当前最好的项目集,然后将*cNewNode*返回给算法*DoCover*(第5~8行)。否则,我们继续使用*cSeedNode*和*gSeedNode*扩展项目集(第9行)。

4 得分模型与分类策略

给定一个预定义的类集$\{C_1, C_2, \cdots, C_M\}$和一个待分类文档$D_w$,首先需要识别类$C_i(1\leqslant i\leqslant M)$和文档$D_w$共享的所有的"公共项目集",即$C_i\cap D_w$,并统计它们在$D_w$中的支持度。文档$D_w$与类$C_i$的相似度就可以通过那些公共项目集得到。

公共项目集的识别和它们的支持度计数是同时进行的。对于D_w中的每一个事务,我们检查是否其任何非空子集匹配了类C_i的类前缀树上的有效节点(即修剪后的有效规则对应的节点)。如果匹配成功,将匹配节点的I_i计数器加1。当处理了所有的事务以后,就找到了$C_i\cap D_w$中的每一个公共项目集,它被保存在类C_i的类前缀树上并与在文档D_w中的支持度相关联。

$C_i\cap D_w$中项目集的出现可以被认为是一个指示器,它表示了D_w中存在对应于项目集的类C_i的特征。这里,关键的问题是如何将每一个公共项目集的指示器的强度与衡量D_w与C_i相似程度的总得分结合起来。

在文[18]中,我们使用了传统的中心向量模型作为得分模型。下面我们提出3种新的得分模型。

如果我们同样看待每一个指示器强度,一个显而易见的得分模型是将公共项目集的数目作为将D_w分到类C_i中的得分。形式化地,得分函数$sim(C_i\leftarrow D_w)$定义如下:

$$sim(C_i\leftarrow D_w)=|C_i\cap D_w| \quad (1)$$

如果我们更多地考虑公共项目集的置信度和支持度。自然地,产生了两个带权的变种:

$$sim(C_i\leftarrow D_w)=\sum I_{conf}\times[(I\in C_i\cap D_w)] \quad (2)$$

$$sim(C_i\leftarrow D_w)=\sum(I_{conf}\times I_i)\times[(I\in C_i\cap D_w)] \quad (3)$$

其中$[I\in C_i\cap D_w]$为*Iverson*常数,即如果项目集I属于$C_i\cap D_w$,它取值1,反之取值0。

一旦未分类文档D_w与所有类的相似度已经被计算了,就可以使用如下分类策略来实现多分类:首先赋予D_w最高相似得分的类的类标签。然后,指定一个称为*minsup*的百分比下限。如果其它的类的相似得分大于最高得分的下限百分

比，D_e 也被赋予该类。

5 实验研究

这一节通过通用数据集上的实验，采用标准的有效性度量方法来研究 SAT-FOIL＋方法的分类有效性。将首先研究重要参数的确定是如何影响 SAT-FOIL＋的分类精确性的，然后与其它的文本分类方法的分类精确性进行对比。我们在 Reuters 数据集上实行多分类。

5.1 度量标准

给定 M 个预定义的类别，分类有效性度量标准中微平均与宏平均的查准率(Precison)和查全率(Recall)的定义如表1所示。

表1 查准率和查全率的宏平均与微平均

	P	R
micro-averaged	$\frac{TP}{TP+FP}=\frac{\sum_{i=1}^{M}TP_i}{\sum_{i=1}^{M}(TP_i+FP_i)}$	$\frac{TP}{TP+FN}=\frac{\sum_{i=1}^{M}TP_i}{\sum_{i=1}^{M}(TP_i+FN_i)}$
macro-averaged	$\frac{\sum_{i=1}^{M}P_i}{M}=\frac{\sum_{i=1}^{M}\frac{TP_i}{TP_i+FP_i}}{M}$	$\frac{\sum_{i=1}^{M}R_i}{M}=\frac{\sum_{i=1}^{M}\frac{TP_i}{TP_i+FN_i}}{M}$

在表1中，TP，TN，FP and FN 分别表示真正例，真负例，假正例，假负例。为了便于比较，我们也使用了平衡点(break-even point，BEP)作为单个度量标准：当查准率 P 等于查全率 R 时，被认为达到了平衡点。通常，很难达到了真正的平衡点，特别是要求不同的类同时达到平衡点，因此，通常采用当 P 和 R 充分接近时 P 和 R 的算术平均值作为插值的平衡点。

5.2 文本数据集与预处理

Reuters-21578数据集：ModApte 划分，它包括9603篇训练文档和3299篇测试文档。自然地，如果一个类的训练文档太少，不能期望分类有很好的效果。因此，许多研究者主要选择了 Reuters 语料中最大的10个分类进行实验研究。我们也采用了这个策略。采用了 jihe 提取的 Reuters 十个最大的类别作为预处理的起点[16]。将该训练集用于多分类任务。对语料所做的数据预处理如下：

将每一篇文档中的单词全部换成小写，所有的由数字字符构成的单词，和长度超过24个字符的过长的单词被过滤掉。句子切分时，采用“.”、“?”、“!”和“;”作为句子切分的分隔符；超过30个词的过长句子，被人为地切分成每句30个单词的多个句子以处理没有结束符的表格的情况。对每一个训练文档，我们进行了去停词[13]、提取词干[13]、编码、分句等预处理工作。

5.3 重要参数设置

·训练集文档最小支持度(trainMinsup)：一篇文本中的最小句子支持数，即频繁项目集必须至少在该文本的 *trainMinsup* 个句子中频繁。在后面的实验中，典型的 *trianMinsup* 取值是：2，3，5，8。

·类最小支持度(catMinsup)：类频繁项目集的最小支持度，即类频繁项目集必须至少在类的 *catMinsup* 个文档中是文档频繁项目集。在后面的实验中，典型的 *catMinsup* 取值是：1，2，3。

·测试集文档最小支持度(testMinsup)：测试集文档匹配的最小支持度，即要求被匹配的项目集至少出现在测试集文档的 *testMinsup* 个句子中。在后面的实验中，典型的 *testMinsup* 取值是：1，2。

·得分模型(scoreModel)：计算相似度的得分模型。在后面的实验中，采用的得分模型有：模型4.1～4.3。

·分类最小支持度(labelMinsup)：分类时，决定分类结果的最小百分比。当进行单分类时，该值为100%。当进行多分类时，典型的 *labelMinsup* 取值范围是60%～80%。

5.4 实验结果

我们使用5.3小节中提到的5个参数的典型值的组合来寻找在三个代表性数据集上的最佳分类效果。

5.4.1 实验训练事务最小支持度 *trainMinsup* 的结果 从表2中，我们可以看到在 *natural trainMinsup* 2时，分类结果最好；随着 *trainMinSup* 的提高，分类效果逐步下降。其中，当 *trainMinSup* 从5提高到8，分类效果急剧下降。实际上，*trainMinSup* 的取值和实验的数据集有很大关系。据我们观察，在 *Reuters* 数据集中，平均每一个训练文本仅仅包含6.7条句子，太高的 *trainMinSups*，例如8，将丢失重要的 DFIs。

表2 *trainMinsup* 不同取值下的最好结果

	SAT-FOIL＋	
trainMinsup	MicroAugBEP(%)	MacroAvgBEP(%)
2	90.3%	83.2%
3	89.7%	82.8%
5	85.2%	73.9%
8	62.8%	62.8%

表3 *catMinsup* 不同取值下的最好结果

	SAT-FOIL＋	
catMinsup	MicroAvgBEP(%)	MacroAvgBEP(%)
1	89.7%	83.7%
2	89.6%	83.0%
3	90.4%	83.2%

5.4.2 实验 *catMinsup* 的结果 从表3中，我们可以看出，SAT-FOIL＋和在 *catMinSup* 取值为2或3的时候得到最好的分类结果。*catMinsup* 取值为1时，实际上表示没有经过类最小支持度修剪。

5.4.3 实验 *testMinsup* 的结果 从表4中，我们可以看出，*testMinsup* 取值为1时，分类效果最好，且比取值为2时，要好得多。因此，我们可以认为，在 Reuters 数据集上，计算分类得分时，不必要求 CFIs 在测试文档中文档频繁。

表4 *testMinsup* 不同取值下的最好结果

	SAT-FOIL＋	
testMinsup	MicroAvgBEP(%)	MacroAvgBEP(%)
1	90.3%	83.2%
2	82.0%	76.3%

5.4.4 实验得分模型的结果 从表5中，我们可以看出，得分模型4.3最好，4.2次之，4.1再次之。得分模型4.1仅仅使用匹配的分类规则计分，得分模型4.2使用匹配的分类规则的 confidence 作为权重计分，得分模型4.3同时利用了两者用于计分：采用 confidence 作为权重，并累加被匹配的分类规则的个数。因此，可以发现 confidence 不仅仅对于训练过程中分类规则修剪是必要的，而且对于分类过程中改善分类得分计算，从而提高分类效果也是有益的。

表5 不同得分模型下的最好结果

	SAT-FOIL+	
scoreModel	MicroAvgBEP(%)	MacroAvgBEP(%)
4.1	82.0%	74.7%
4.2	87.2%	80.4%
4.3	90.3%	83.2%

5.4.5 Reuters 数据集的最佳结果

在表6中，我们使用当前 SAT-FOIL+最好的分类结果与 SAT-FOIL 以及其它知名的方法进行比较。列“SAT-FOIL+”的参数是：*trainMinsup*=2，*catMinsup*=3，*testMinsup*=1，*labelMinsup*=70%或80%。列“SAT-FOIL”：*trainMinsup*=2，*catMinsup*=3，*testMinsup*=1，*labelMinsup*=70%或87%。

表6 SAT-FOIL+在 Retures10个类上的最好结果

BEP	SAT-FOIL+ labelMinsup		SAT-FOIL+ Minsup		ARC-BC δ=50		Bayes	Rocchio	C4.5	k-NN	linear SVM
	70%	87%	70%	80%	10%	15%					
acq	85.8	87.2	91.8	92.7	90.9	89.9	91.5	92.1	85.3	92.0	93.6
coTn	75.2	67.9	70.7	69.4	69.6	82.3	47.3	62.2	87.7	77.9	90.3
crude	78.5	82.1	90.1	90.5	77.9	77.0	81.0	81.5	75.5	85.7	88.9
earn	91.9	94.7	95.0	96.2	92.8	89.2	95.9	96.1	96.1	97.3	98.0
grain	88.6	85.3	89.3	86.4	68.8	72.1	72.5	79.5	89.1	82.2	94.6
interest	67.1	67.8	78.9	76.3	70.5	70.1	58.0	72.5	49.1	74.0	77.7
money-fx	69.1	73.8	84.0	82.7	70.5	72.4	62.9	67.6	69.4	78.2	74.5
ship	81.7	70.7	83.1	80.9	73.6	73.2	78.7	83.1	80.9	79.2	85.6
trade	78.6	78.1	86.9	86.7	68.0	69.7	50.0	77.4	59.2	77.4	75.9
wheat	79.1	62.2	72.7	70.4	84.8	86.5	60.6	79.4	85.5	76.6	91.8
micro-avg	84.9	86.0	90.1	90.3	82.1	81.8	72.0	79.9	79.4	82.3	92.0
macro-avg	79.6	77.0	84.3	83.2	76.7	78.2	65.2	79.1	77.8	82.1	87.1

ARC-BC 是目前已知的最好的文档级的关联文本分类算法，从表6中，可以看到 SAT-FOIL+的分类效果可比于那些知名的算法；除类“corn”和“wheat”外，在其它的8个类上，SAT-FOIL+比 ARC-BC 的分类效果要好得多。与 SVM 相比，10个类中有5个类的分类效果要好一些。SAT-FOIL+在类“corn”和“wheat”中效果较差，而在类“grain”中的效果较好。事实上，类“corn”和“wheat”都是类“grain”的子类，较多的文档都正确地分到了父类中，但是并没有分到任何一个子类中。一个可能的原因是父类中含有某些子类没有的特性(项目集)，在分类时，这些特性增加了父类的得分。这也许可以使用降低 *labelMinSup* 以便将更多的文档分入到子类中来解决。我们将在以后的工作中研究对不同的类使用不同的 *labelMinSup* 以改善分类效果。类似的情况还出现在类“interest”和“money-fx”。从表6中可以看到，采用新得分模型的 SAT-FOIL+的分类结果比采用中心向量模型的 SAT-FOIL 的分类结果要好得多。

总结 CBA 方法首次集成了分类和关联规则。然而，它仅仅进行数值数据的单分类。如文[15]中指出的，分类和关联规则挖掘的重要差异是分类要避免过配(overfitting)/低配(underfitting)。通常一些训练集的覆盖策略被用来选择发现的分类规则，而我们使用了 FOIL 进行自适应的选择。

文[14]提出了2种 ARC 算法来执行基于关联的文本分类。为了解决过多的可用规则将太多的类赋予一篇未分类的文档的问题，作者介绍了优势因子的概念。优势因子是在对一个文档进行分类时，应用规则中最重要的类的规则所占的比例。当优势因子被定义成百分比的时候，只有那些含有的规则数超过可用规则的数目乘以优势因子的类被提取，将最靠前的 K 个类标签赋予文档。该算法是将整个文档作为一个单独的事务。

除了基于关联的文本分类，频繁项目集也可以用于文本聚类。在文[11]中，提出了 FIHC 算法。其思想是文档集中的每个主题都存在一些频繁项目集，而不同主题之间不会共享很多的频繁项目集。我们的类频繁项目集的概念是受前述思想的启发：属于同一个类的文档应当共享许多频繁项目集。作者还介绍了频繁项目集会明显地减少文档的维度。注意到 FIHC 仍然是将整个文档看作一个事务。

在本文中我们提出 SAT-FOIL 的改进算法 SAT-FOIL+，它采用了全新的得分模型，大量的实验证明新的模型明显优于 SAT-FOIL 所采用的中心向量模型。SAT-FOIL+的分类效果已被证明同那些著名的分类算法是可比的，并且远远好于以前的基于文档级关联的分类方法。

下一步的工作是将我们的 SAT-FOIL+方法运用到更为复杂的数据集比如20NG 数据集[17]上，并且实验使用一个段落的首句与尾句来取代该段落的办法加快算法的运行。另一个可能的方向是将 SAT-FOIL+的基本思想运用到其他以前将文档视为一个事务的分类方法中，比如 Large Bayes 或 FIHC。

参考文献

1 Sebastiani F. Machine Learning in Automated Text Categorization. ACM Computing Surveys, 2002,34(1):1～47

2 Dumais S, Platt J, Heckerman D, Sahami M. Inductive Learning Algorithms and Representations for Text Categorization. CIKM98

3 Agrawal R, Srikant R. Fast Algorithms for Mining Association Rules. In:Proc of the 20th Very Large Data Bases,1994

4 Liu B, Hsu W, Ma Y. Integrating classification and association rule mining. In:SIGKDD, 1998.80～86

5 Antonie M, Zaiane O R. Text Document Categorization by Term Association. In: Proc. of IEEE Intl. Conf. on Data Mining, 2002

6 Joachims T. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In: European Conf. on Machine Learning, 1998.137～142

7 Meretakis D, Fragoudis D, Lu H, Likothanassis S. Scalable Associ-

ation-based Text Classification. In:Proc. of ACM Int. Conf. on Information and Knowledge Management, 2000
8 Bekkerman R,El-Yaniv R,Tishby N,Winter Y. Distributed Word Clusters vs. Words for Text Categorization. Journal of Machine Learning Research, 2003,3:1183～1208
9 Quinlan J R,Carneron-Jones R M. FOIL:A Midterm Report. In: Proc. European Conf. Machine Learning,1993. 3～20
10 Baeza-Yates R, Ribeiro-Neto B. Modern Information Retrieval. Addison-Wesley, 1999
11 Fung B C M, Wang K. Hierarchical Document Clustering Using Frequent Itemsets. In:Proc. of SIAM Intl. Conf. on Data Mining, 2003
12 Borgelt C. Efficient Implementation of Apriori and Eclat. In:Proc of the first Workshop on Frequent Itemset Mining Implementations, 2003
13 Frakes W B, Baeza-yates R. Information Retrieval: Data Structures and Algorithms, Prentice-Hall, 1992
14 Zaiane O R,Antonie M. Classifying Text Documents by Associating Terms with Text Categories. In:ADC, 2002. 215～222
15 Freitas A A. Understanding the Crucial Differences Between Classification and Discovery of Association Rules-A Position Paper. SIGKDD Explorations,2000,2(1):1～5
16 http://www.jihe.net/datasets.htm
17 The 20 Newsgroups Dataset. http://people.csail.mit.edu/u/j/jrennie/public_html/20Newsgroups/
18 李曲,冯剑琳,邹晶,冯玉才. SAT-FOIL: Sentence as Association Transaction for Text Classification. 已投稿

(上接第196页)

等。通过这些操作,非结构化数据可以近似地转化为结构化数据,然后再进行数据挖掘,这是非结构化数据挖掘的必由之路。

邸凯昌提出的四维空间知识发现状态空间也可归纳进来,也是本统一框架理论的一个特例。因为空间尺度维代表的是不同的分辨率的变化,它与图像的放大、缩小操作(使得面状目标变成点状目标),以及分辨率处理是一致的,也属于特征的一种特殊变换,因此,空间尺度维可以合并到特征维 F 轴上来。

3 应用

知识发现状态模型已经为 Web 文挖掘提供了理论指导,并取得了很好的研究效果[5]。Web 文本挖掘系统界面如下图5、图6所示,Web 文本挖掘原型系统的开发环境是 Windows 操作系统,数据库管理系统使用了微软的 SQL Server 数据库,开发工具使用了面向对象的集成开发工具 Delphi6.0。考虑到文章篇幅所限,Web 文本数据挖掘应用过程简述如下:

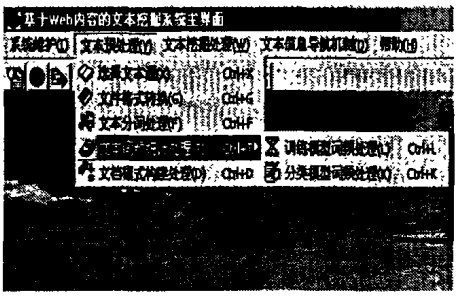

图5 Web 文本挖掘系统主界面(1)

(1)文本数据预处理过程:包括特征表示、文本信息的分词预处理、特征提取等过程。其中特征表示是指以一定的特征项(如单字或词条等)来代表文档信息;文本信息的预处理主要包括英文文档的词干处理和中文文档的词条切分;特征提取是指特征表示中词条 T 的选取,是挖掘特征共性与规则的提取过程。

(2)文本数据的知识发现算法:基于模式的文本挖掘是一个发现新模式或对模式进行某种确证的过程。通过模式矢量,可同文本分类、聚类、关联模式等收敛型的知识发现算法及预测、时序等发散型的知识发现算法相结合,来完成对于各种文本数据的知识发现。

(3)模式的评价:文本模式的评价方法将构造特定的评价指标。该指标的选择应该符合评价的主客观标准,采用定量的方式来评估结果模式集中有效的、新颖的、潜在可用的及最终可理解的模式。所采用的评价指标为分类正确率、查全率、查准率、综合分类率及平均准确率等指标。

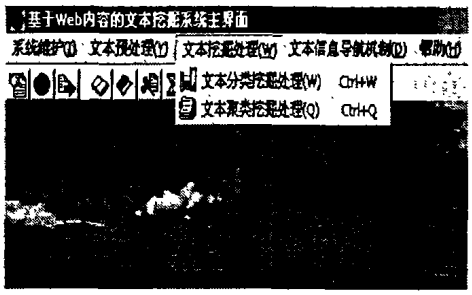

图6 Web 文本挖掘系统主界面(2)

(4)模式的解释与呈现:Web 文本挖掘系统最终挖掘出来的模式能够用可视化的方式进行显示,同时对用户提供概念导航机制的功能,使用户有效、快速地浏览和获取信息。

结论 通过分析,可以得出:

(1)UMKDSS 刻画了结构化数据挖掘与非结构化数据挖掘之间的关系,指出了复杂类型数据挖掘的重点和难点在于它的特征处理,即如何提取特征并降低特征空间的维数,使其近似转化为结构数据,然后再进行数据挖掘。

(2)UMKDSS 既包含结构化数据挖掘,又包含复杂类型数据挖掘,并将结构化数据挖掘与复杂类型数据挖掘有机地统一起来,成为知识发现领域的统一框架理论,对 KDD 的发展有积极的指导意义。

参考文献

1 李德毅. 发现状态空间理论[J]. 小型微型计算机系统. 15(11)
2 邸凯昌. 空间数据发掘与知识发现[M]. 武汉大学出版社,2001. 20～23
3 Yang Bingru. Double-Bases Cooperating Mechanism in KD(D&K) System[C]. IC-AI'99,1999 (USA)
4 Yang Bingru. Structrue of Knowledge Discovery Based on Double-Base Cooperating Mechanism[J]. Data Mining and Knowledge Discovery,1999
5 唐菁. 基于知识发现内在机理的 Web 文本挖掘结构模型与算法研究[D]. 北京:北京科技大学,2003