基于句子级的最大频繁序列的文本分类*)

邹 晶 冯剑琳 李 曲 王元珍
(华中科技大学计算机学院 武汉 430074)

摘 要 本文提出了一种新的文本分类方法。这种方法将一篇文本的一个句子看作一个事务,一个段落看作是一个序列,则一篇文本表示成一个序列的集合。我们从每篇训练文本中挖出最大频繁序列用以表示这篇文本,这种表示方法可大大提高训练及分类速度,同时也可以几乎不损失分类精度。在数据集 Reuters-21578[1] 上的大量实验证明这种方法要远远好于其他的文本级的基于关联的分类方法。

关键词 文本分类,句子级,最大序列,频繁序列

Text Classification Based on Sentence-level Maximal Frequent Sequence

ZOU Jing FENG Jian-Lin LI Qu WANG Yuan-Zhen
(Department of Computer Science, Huazhong University of Science and Technology, Wuhan 430074)

Abstract In this paper, we present a novel text classification method . It views a sentence as an association transaction, and a paragraph as a sequence, then a document becomes a set of sequneces. We find maximal frequent sequences from each training document to present it, so the training and classification speed can be improved greatly. The effectiveness of this method has been demonstrated comparable to well-known alternatives and much better than current document-level words association-based methods on the Reuters corpus.

Keywords Text classification, Sentence-level, Maximal sequence, Frequent sequence

1 引言

文本分类是根据一个文本的内容将它分到一个或多个预先定义好的类。它的应用如邮件分类或垃圾邮件的过滤,网页的搜索及浏览等。目前为止,已经提出了多种自动文本分类的方法,如 Naïve Bayes[3],决策树[4],k-NN[5],Rocchio[6],神经网络方法[7],支持向量机 SVM[8]等。以前的基于关联规则的文本分类方法大都是挖掘文档级的频繁项集[9~12],即将一个文本看作一个关联事务,挖掘出那些同时出现在一篇文本中的某些词的组合。但是在一个文本中,句子才是基本的语义单元。在一个句子中,某几个词同时出现;在一段中,某几个词在第一个句子中出现,另外一些词在接下来的一个句子中出现,这样才是有意义的。直觉上,一个序列可以这样理解:在一段的依次的几个句子中,某些词出现在一个句子中,另外一些词出现在随后的某个句子中,还有一些词出现在接下来的某个句子中,依次这样下来,得到的就是一个序列。如果这样的序列出现在一个文本的 2 个或 2 个以上的段中,说明它是频繁出现的,它可以代表这个文本的一个方面。

文[2]中提出的 SAT-MOD 算法,是一种将句子作为一个事务的基于关联规则的文本分类方法。在这篇文章中还提出了 MODFIT 的修剪规则的方法。实验证明,SAT-MOD 的分类效果要远远好于基于关联的文本级的文本分类方法。

本文第 2 部分描述怎样用一些文档最大频繁序列表示一个文本。在第 3 部分中,我们详细描述怎样用这些序列构造一个分类器及分类策略。第 4 部分我们分析实验结果。最后总结全文。

2 挖掘文档最大频繁序列

2.1 基本定义

根据文[13]的定义,一个项目集是一个非空的项目的集合,一个序列是一些项目集的有序的列表。文[13]中一个典型的序列的例子是超市购物的例子。一件商品是一个项目,每个项目有一个唯一的整数编号,一个用户的一次购物为一个项目集,一个用户的所有购物事件构成一个序列。如用户 A 在 2004 年 5 月 5 日购买了 1,3,5,在 2004 年 5 月 8 日购买了 2,3,6,2004 年 5 月 18 日购买了 1,6,则按照购买的时间顺序,用户 A 构成一个序列{(1,3,5),(2,3,6),(1,6)},所有的用户构成一个序列数据库。

我们用$\{S_1,S_2,\cdots,S_n\}$表示一个序列,$S_i(1\leqslant i\leqslant n)$为项目集。序列 $A\{A_1,A_2,\cdots,A_n\}$ 包含于序列 $B\{B_1,B_2,\cdots,B_m\}$,如果存在一些整数 $i_1<i_2<\cdots<i_n$,使得 $A_1\subseteq Bi_1,A_2\subseteq Bi_2,\cdots,A_n\subseteq Bi_n$,也可以称作序列 A 是序列 B 的子序列。如果一个序列不包含于任何其他的序列中,我们称它为最大序列。

在本文中,我们将一篇文本看作一个序列数据库,一个段落看作一个序列,一个句子看作一个项目集,一个单词看作一个项目。一个序列在文档 D 中的支持度为 S_D,如果这个序列包含于 D 中的 S_D 个段落中。

2.2 文档最大频繁序列的挖掘

如果一个序列在文本 D 中的支持度不小于 S_D,则我们称之为 D 中的频繁序列,S_D 为用户给定的文档最小支持度。如果一个频繁序列不包含在从这个文本中挖出来的其他的频

*)Supported by the Natural Science Foundation of Chongqing Province of China under Grant No. 8721(重庆市自然科学基金) and Chinese Doctor Stie research grant No. 20030487032(中国博士点基金)。邹 晶 硕士研究生,主要研究领域为文本挖掘;冯剑琳 博士,主要研究领域为 OLAP 和文本挖掘;李 曲 博士研究生,主要研究领域为文本挖掘;王元珍 教授,博士生导师,主要研究领域为数据挖掘。

繁序列中，则称之为文档最大频繁序列。这部分所要做的就是挖出文档最大频繁序列用以表示此文本。

我们利用文[14]中的算法来挖出所有的文档频繁序列后，最后自己加上了一步用于找出最大序列。

在下面的实验中，我们作如下比较：将一篇文本中每个句子作为一个项目集，每个文本作为一个序列，所有属于一个类的文本就构成了一个序列数据库，在这样的数据库中来挖掘频繁序列，我们称之为文本级的。另外一种就是本文中所提出的，将一篇文本看作一个序列数据库，一个段落看作一个序列，一个句子看作一个项目集，我们称之为段落级的。相较于文本级的表示方式，段落级的有以下两点优势：

1. 当类的大小不一的时候，如果用文本级的表示方法，挖掘频繁序列的时候最小支持度很难统一确定。如果设大了，小类挖出的频繁序列会非常少，训练会很不充分；如果设小了，大类会挖出大量的频繁序列，不仅有些会是冗余的，而且会大大降低训练的速度。而段落级由于是限定在文本内的，因此参数设置相对较简单，可统一设为一个整数(如 2)。

2. 如果将整个文本看作一个序列，则其子序列非常多，而本文中所用的分类方法是要与所有子序列匹配的，所以如果子序列太多的话会大大降低分类速度。例如一篇测试文本有 100 个单词，如果用整个文章作为一个序列的话，则最多有 2^{100} 个子序列，但是如果将它分成 10 段，每段 10 个单词的话，则最多有 $10 * 2^{10}$ 个子序列。

3 分类器的构造

3.1 类频繁序列

哪些序列可以表示一个类呢？我们要找出那些同一个类下不同文档之间的“公有文档主题”，它可以代表整个“类主题”的一个方面，我们称之为类频繁序列。

如果序列 S 在类 C 的 S_C 个文档中是频繁的，则 S 在 C 中的支持度为 S_C。如果 S 在文档 D 中是文档频繁的，则 D 称为 S 的支持文档。

如果序列 S 在类 C 中的支持度不小于用户指定的最小类支持度，则 S 为 C 的类频繁序列。自然地，最小类支持度应该设为 2 以保证一个类频繁序列至少在类的 2 篇文本中是频繁的。然而如果没有充足的训练文档，单独一篇文档也可能代表了“类主题”的一个方面，因此，我们将类的最小支持度默认值设为 1。

类似文[2]中规则的置信度的定义，我们定义序列的置信度。

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

3.2 序列树的构造及修剪

参考文[14]中字典序树的构造，我们构造一个序列树来收集类频繁序列。序列树如图 1 所示。

在序列树中，除根节点外，每个节点表示一个序列，每一条边标记一个项目(假定项目按它的编码大小有个顺序)。序列树按如下方法构造：如果 n 是树中的一个节点，则它的孩子节点 m 是在 n 所代表的序列的末尾扩展这条边所标记的项目所得到的序列，n 一定是 m 的子序列。序列树中的每个序列要么是一个序列扩展的序列，要么是一个项目集扩展的序列。一个序列扩展序列是在它的父节点对应的序列末尾添加一个新的项目集所得到的，这个新的项目集由单个的项目组成。一个项目集扩展序列是在它的父节点所表示的序列中的最后一个项目集的末尾添加一个新的项目所得到的，这个新的项目的编号大于最后一个项目集中的所有项目的编号。例如图 1 中序列{1,1}的两个序列扩展孩子分别为{1,1,1}和{1,1,2}，它的一个项目集扩展孩子为{1,(1,2)}。图 1 中，序列扩展孩子用实线标出，项目集扩展孩子用虚线标出。

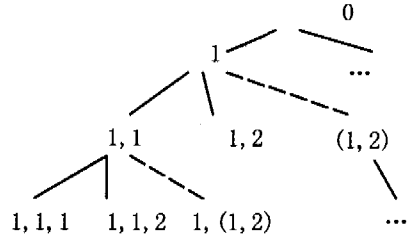

图 1 序列树

由以上定义可知，对于树中的任一节点，它可能有两种类型的孩子节点，一类为序列扩展的孩子，一类为项目集扩展的孩子。每一个节点包含 3 个计数器：I_c，I_{cmf} 和 I_s。我们使用 I_c 和 I_{conf} 分别表示结点所对应的序列 S 的类支持度和置信度。计数器 I_s 用于分类阶段，统计序列 S 在新文档 D_u 中的文档支持度。

给定 M 个预定义的类，我们利用属于每个类的文档最大频繁序列为其构造一棵序列树。同时我们构造一棵全局树以收集所有训练文档中的最大频繁序列。这棵全局树用于计算 S_{tot}，它表示在整个训练文本集中一个序列所覆盖的不同支持文档的总数。

一旦全部 M 棵序列树及全局树构造完毕，我们使用 3.1 节中置信度的定义计算每一棵序列树中每个节点的置信度(I_{conf})。

然后我们使用 MODFIT 算法[2]来修剪每棵序列树。直觉上，MODFIT 修剪等价于沿着序列树的根结点开始的路径扩展空的序列，每次扩展一个项目，序列扩展或者项目集扩展，直到序列的置信度不再有任何增长。

我们在 2.2 节中挖掘最大序列的时候，判断一个序列是不是最大的，要求找出它的所有的子序列，也是用的这样的一个序列树的结构。

3.3 分类

给定一个预定义的类的集合$\{C_1, C_2, \cdots, C_M\}$和它们对应的类序列树的集合$\{CST_1, CST_2, \cdots, CST_M\}$，定义每棵 $CST_i (1 \leqslant i \leqslant M)$ 和待分类文档 D_u 之间共享的“公共序列”为类 C_i 和 D_u 的类交集(记为 $CST_i \cap D_u$)，我们用 I_s 来记录每个“公共序列”在 D_u 中出现的次数，即记录 D_u 中有多少个段(序列)包含此序列。文档 D_u 在类 C_i 中的得分如下计算：

$$score(C_i \mid D_u) = \frac{\Sigma I_{cmf} \times I_s}{\Sigma I_s} \times [S \in CST_i \cap D_u] \times \frac{|CST_i \cap D_u|}{\max_{i \in \{1,\cdots,M\}} |CST_i \cap D_u|}$$

其中$[S \in CST_i \cap D_u]$为 *Iverson* 常数，即如果序列 S 属于 $CST_i \cap D_u$，它取值 1，反之取值 0。

每到来一个新的文档，我们计算它与每个类的得分，如果是单分类，则将其分到得分最高的那个类中，如果是多分类，则将其分到得分高于最高得分与 labelMinsup 乘积的所有类中，其中 labelMinsup 为用户指定的分类百分比。

4 实验研究

4.1 数据集预处理

Reuters-21578 数据集[1]：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，我们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。

对于训练集和测试集中的文本，我们只取题目和主体部分。每一篇文本中的单词全部被换成了小写，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。一个单词在一个句子中的多次出现只被计为一次。句子切分时，我们采用“.”、“?”、“!”和“;”作为句子切分的分隔符，并加入了缩写与点号混淆问题的处理；段落切分时，我们采用“.”加回车符作为段落切分的分隔符。对每一个文档，我们进行了去停词，提取词干、编码，分句/分段等预处理工作。

4.2 度量标准

对于多分类，标准的分类有效性的度量为 microaveraged 和 macroaveraged precision(P)和 recall(R)[15]。为了比较方便，我们使用称之为 break-even point 即 BEP[16]的衡量方法。

4.3 实验分析

我们先来比较本文中所提出的分类方法(SAT-Pattern)与其他的一些分类方法，如表 1 所示。

表 1 REUTERS 最大的 10 个类上的 BEP

BEP	SAT-Patter labelMinsu	ARC-BC δ=50		Bayes	Rocchio	k-NN	LinearSVM
	65%	10%	15%				
acq	90.2	90.9	89.9	91.5	92.1	92.0	93.6
corn	57.6	69.6	82.3	47.3	62.2	77.9	90.3
crude	85.4	77.9	77.0	81.0	81.5	85.7	88.9
earn	93.1	92.8	89.2	95.9	96.1	97.3	98.0
grain	86.7	68.8	72.1	72.5	79.5	82.2	94.6
interest	79.0	70.5	70.1	58.0	72.5	74.0	77.7
money-fx	86.9	70.5	72.4	62.9	67.6	78.2	74.5
ship	84.0	73.6	73.2	78.7	83.1	79.2	85.6
trade	81.6	68.0	69.7	50.0	77.4	77.4	75.9
wheat	75.1	84.8	86.5	60.6	79.4	76.6	91.8
micro-avg	88.2	82.1	81.8	72.0	79.9	82.3	92.0
macro-avg	81.95	76.74	78.24	65.21	79.14	82.05	87.10

文[10]中所提出的 ARC-BC 算法是目前最好的文本级的，基于关联的文本分类方法。从表 1 可以看出，除了线性 SVM 外，我们的方法(SAT-Pattern)要远远好于 ARC-BC 及其他的几种方法。表 1 里的其他几种方法的结果是从文[17]中得出的。

接下来，我们做了一组实验，比较文本级与段落级的文本表示方法的分类有效性。

用文本级的表示方法时，挖掘的序列数据库为属于这个类的所有训练文本，由于类的大小不一，挖掘的最小支持度难以统一设置。只有人为地为每个类配置一个不同的最小支持度，使得每个类挖出的序列数大致相等。

采用段落级的表示方法时，挖掘的序列数据库为每篇训练文本，可统一设置文档最小支持度为 2，即要求一个序列至少在一个训练文本的 2 个段落中出现。对于较长的文本，段落数较多(超过 10)，可将文档最小支持度设为 3，以免挖出太多的序列。这样相对于文档级的参数确定就简单多了。

两种表示方法的分类结果及分类时间(包括输入输出的时间，时间单位为秒)如表 2 所示。

表 2 文本级与段落级的比较

表示方法	Micro-avg	Macro-avg	分类时间
文本级	89.2%	81.3%	713
段落级	88.2%	81.9%	70

由表 1 可以看出，段落级的表示方法与文本级的表示方法分类效果相差无几，但在分类时间上，文本级的分类速度远远慢于段落级的，具体原因在前面已经分析过了，实验结果也证明确实如此。

下面我们做了一组实验，来比较分别用最大序列(MAX)和非最大序列(NON-MAX)表示文本的情况下，训练及分类速度上的不同。

由表 3 可以看出，在分类精度上这两种表示方法相差无几。在训练速度上，最大序列的明显要好于非最大序列的，时间只为其十分之一，因为在用最大序列表示的情况下，读入的序列少一些，且剪枝后留下的有效的序列也少些，这样另一方面也可以提高分类的速度。在程序运行的时候，很明显地可观察到所消耗的 CPU 及内存资源也要远远少于非最大序列的。另外，由于只输出最大序列，对于大数据集的挖掘的可行性也大些。

表 3 最大序列与非最大序列的比较

表示方法	Micro-avg	Macro-avg	训练时间	分类时间
NON-MAX	88.3%	82.5%	207	100
MAX	88.2%	81.9%	19	70

以上实验中，labelMinsup 的取值为 65%，最小文档支持度为 2，最小类支持度为 1。实验在一台 CPU 为 PⅢ800，内存为 512M 的机器上做的。

总结 本文中提出了一种新的文本分类方法，用最大频繁序列来表示一个训练文本而进行分类，这种表示是段落级的，因为考虑词在句子间的出现顺序也是有意义的，实验结果也证明了这样是有效的。接下来，我们可以利用更多语言学的知识来改善段落和句子切分，并且在其他的一些数据集上的实验来验证这种方法的有效性。

参 考 文 献

1 The Reuters-21578 Dataset. http://www.daviddlewis.com/resources/testcollections/reuters21578/

2 Feng Jianlin, Liu Huijun, Zou Jing. SAT-MOD: Moderate Itemset Fittest for Text Categorization. WWW2005 in Chiba. Invited Poster Paper, 2005. 1054～1055

3 Robertson S E, Sparck Jones K. Relevance weighting of search terms. J. Amer. Soc. Inform. Sci. 27, 3, 129～146. Also reprinted in Willett [1988], 143～160

4 Cohen W W, Hirsh H. 1998. Joins that generalize: text classification using WHIRL. In: Proc. of KDD-98, 4th Intl. Conf. on Knowledge Discovery and Data Mining (New York, NY, 1998), 169～173

5 Yang Y. Expert network: effective and efficient learning from human decisions in text categorisation and retrieval. In: Proc. of SIGIR-94, 17th ACM Intl. Conf. on Research and Development in Information Retrieval (Dublin, Ireland, 1994), 13～22

6 Hull D A. Improving text retrieval for the routing problem using

latent semantic indexing. In: Proc. of SIGIR-94, 17th ACM Intl. Conf. on Research and Development in Information Retrieval (Dublin, Ireland, 1994), 282~289
7 Lam S L, Lee D L. Feature reduction for neural network based text categorization. In: Proc. of DASFAA-99, 6th IEEE Intl. Conf. on Database Advanced Systems for Advanced Application (Hsinchu, Taiwan, 1999), 195~202
8 Joachims T. Text categorization with support vector machines: learning with many relevant features. In: Proc. of ECML-98, 10th European Conf. on Machine Learning (Chemnitz, Germany, 1998), 137~142
9 Agrawal R, Srikant R. Fast Algorithms for Mining Association Rules. In: Proc. of VLDB, 1994. 487~499
10 Antonie M, Zaiane O R. Text Document Categorization by Term Association. In ICDM, 2002. 19~26
11 Han J, Pei J, Yin Y. Mining Frequent Patterns without Candidate Generation. In SIGMOD, 2000. 1~12
12 Brutlag J D, Meek C. Challenges of the Email Domain for Text Classification. In: Proc. of ICML, 2000. 103~110
13 Agrawal P, Srikant R. Mining Sequencial Patterns: [Research Report]. 1995
14 Ayres J, Gehrke J. Sequential PAttern Mining using A Bitmap Representation, SIGKDD 02
15 Sebastiani F. Machine Learning in Automated Text Categorization. ACM Computing Surveys, 2002, 34(1): 1~47
16 Grahne G, Zhu J. Efficiently Using Prefix-trees in Mining Frequent Itemsets. In: Proc. FIMI, 2003
17 Dumais S, Platt J, Heckerman D, Sahami M. Inductive Learning Algorithms and Representations for Text Categorization. In CIKM98. 148~155

(上接第 212 页)

虽然 FCA 有着很强的数学基础，从格中能很方便地给出一些隐含的概念供选择，但是，当本体的应用领域非常复杂时，相应地建立的概念格必将很复杂。这样复杂的格结构将淹没有用信息，从而又为新概念和关系的选择带来难度。

结束语 目前，FCA 用于本体的构建处于刚刚起步的阶段，在实际应用过程中还存在许多的问题，但是 FCA 为构建领域本体这一难题提供了新的解决思路。随着 FCA 中的概念更合理地同本体中的概念联系起来，且更好地同自然语言理解、机器学习等领域的方法相结合，更完善的本体开发工具的出现，我们有理由相信领域本体的构建将不再困难，构建的领域本体定能更好地表达领域并为之服务。

表 2 四种方法的比较分析

		Philipp Cimiano	GuTao	Marek Obitko	Hele-Mai Haav
概念及其关系获取		NLP	NLP/手动	NLP/手动	NLP
背景形式	对象	领域词汇（名词）	类	领域实体	领域文本的编号
	属性	领域词汇（动词）	槽	实体属性	领域词汇集
本体表示		格	Protégé 模型	三元组	格
优点		实现了本体的自动构建；易于实现本体的更新；通过把相同上下文的名词处理为相同的词来解决同义词问题；提供了一套本体的评价方法。	自开发的 FcaTab 插件可自动从领域概念和关系得到形式背景，结合领域专家的参与实现半自动的领域本体构建；可消去分类结构中概念的冗余，并得需要的概念。	提供了分布式的本体编辑环境；按属性进行分类，克服了当前分类方法存在的问题；这种方法构建的本体适用于知识交换。	自动构建领域的形式化本体；实现了本体的逻辑表述，从而易于本体的推理和验证；易于实例化本体和检索本体实例；适用于领域文本内容比较短的情况。
缺点		没有考虑词的多义情况；概念分类相对单一，只考虑了 is-a 关系。	对于属性值是多值的情况，必须先转换成单值才可以使用 FcaTab。	整个过程是通过添加或删除概念和属性调整这样一个不断的迭代的过程，所以到底需要添加或删除哪些内容难以把握，而且具体到什么时候结束也不容易确定。	需要对缩减后的格中的概念命名，并且将概念和具体的文档做映射，这两项任务都是不容易做到的。

参 考 文 献

1 Uschold M. Ontologies Principles, Methods and Applications. Knowledge Engineering Review, 1996, 11(2)
2 Gruninger M, Fox M S. Methodology for the Design and Evaluation of Ontologies. Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95, Montreal, 1995
3 Fernandez M, Gomez-perez A, Juristo N. Methontology: From Ontological Art Towards Ontological Engineering. AAAI-97 Spring Symposium on Ontological Engineering, Stanford University, 1997
4 http://cui.unige.ch/~hilario/icdm-01/DM-KM-Final/Volz.pdf
5 Cimiano P, Staab S, Tane J. Automatic acquisition of taxonomies form text: FCA meets NLP. In: Proceeding of the International Workshop on Adaptive Text Extraction and Mining, 2003
6 Cimiano P, Staab S, Tane J. Deriving concept hierarchies from text by smooth formal concept analysis. In: Proc. of the GI Workshop "Lehren Lernem-Wissen-Adaptivitat" (LLWA), 2003
7 Gu Tao. Using Formal Concept Analysis for Ontology Structuring and Building. ICIS, Nanyang Technological University, 2003
8 http://protege.stanford.edu
9 http://sourceforge.net/projects/conexp
10 Marek O, et al. Ontology Design with Formal Concept Analysis In: Snasel V, Belohlavek R, eds. Concept Lattices and their Applications, Proceedings of the 2nd International CLA Workshop, TU of Ostrava, 2004
11 Haav H M. An Application of Inductive Concept Analysis to Construction of Domain-specific Ontologies. In: Thalheim B, Fiedler G, eds. Emerging Database Research in East Europe. Proceedings of the Pre-conference Workshop of VLDB 2003, Computer Science Reports, Brandenburg University of Technology at Cottbus, 2003. 63~67
12 Haav H M. A Semi-automatic Method to Ontology Design by Using FCA. In: Snasel V, Belohlavek R, eds. Concept Lattices and their Applications. Proceedings of the 2nd International CLA Workshop, TU of Ostrava, 2004. 13~25
13 Noy F N, McGuinness D L. Ontology Development 101: A Guide to Creating Your First Ontology: [Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880]. 2001