文章编号:1008-1534(2009)06-0479-05

基于设计模式的文本分类系统

彭清华[1],朱陈琳[1],范大航[2]

(1.江西省南昌县供电公司,江西南昌 330200;2.石家庄经济学院现代教育技术中心,河北石家庄 050031)

摘 要:基于设计模式设计了一个分类系统,对于分类系统中的数据结构、流程、分类系统的对外接口等问题进行了探讨,提出了"分类任务"的概念,并设计了一个"分类语言"模型。该方案具有简单、灵活及通用性强的特点,对于设计、实现分类系统具有一定的参考价值。

关键词:设计模式;文本分类;分类语言

中图分类号:TP391 **文献标识码**:A

Text classification system based on design pattern

PENG Qing-hua[1], ZHU Chen-lin[1], FAN Da-hang[2]

(1. Electric Company of Nanchang County of Jiangxi Province, Nanchang Jiangxi 330200, China; 2. Modern Education Technology Center, Shijiazhuang University of Economics, Shijiazhuang Hebei 050031, China)

Abstract: This paper designed a text classifying system based on design pattern. The data structure, working flow and external interface of the a system were discussed. A concept named "classifying task" and a "classifying language" model were proposed. The flexible and all-purpose scheme is easy to realize. It is useful for designing and implementing a classification system.

Key words: design pattern; text classification; classifying language

自动文本分类指对自然语言文本按照预定的主题类别进行自动分类过程。它主要应用于信息检索、机器翻译、自动文摘、信息过滤等。近年来,本文分类技术已经逐渐与搜索引擎、信息推送等信息处理技术相结合,有效地提高了信息服务的质量[1]。

目前,人们对分类系统的研究焦点主要集中在分类算法上,而对于分类系统的设计等许多基本问题缺乏统一的论述。笔者提出了"分类任务"的概念,并设计了一个"分类语言"模型。

1 系统框架及基准问题描述

1.1 系统框架设计

众所周知,一个完整的分类任务包括数据集定义、分词、特征提取、分类器训练、分类器测试及分类器的使用等过程。笔者设计的分类系统将这些过程分布在2层架构中:用户层和核心层,见图1。用户层的主要功能包括:1)提供一种分类语言;2)向核心层提交分类语言脚本;3)引用核心层对象。核心层的主要功能包括:1)检查分类语言脚本的合法性并加以执行;2)根据用户定义的参数进行特征选择及分类器训练。

笔者探讨的分类系统是独立于语种的,具有良好的扩展性能及对外使用接口。

1.2 基准问题描述

在实现分类系统时,会存在几个对分类系统性能产生一定影响的基准问题[2]:1)停用词表问题;2)采用向量空间模型的时候,是为每一个类别定义一个向量空间,还是对所有的类别定义一个统一的向

收稿日期:2009-05-21;修回日期:2009-09-14
责任编辑:陈书欣

作者简介:彭清华(1974-),女,江西南昌人,工程师,主要从事电力、计算机应用方面的研究。

量空间？笔者在实验中发现，两者区别不大；3）词干问题；4）分词问题；5）向量空间的大小问题；6）阈值问题；7）特殊文本的预处理问题，如对网页中标签或脚本的处理；8）训练集的划分问题等。

从实验及有关文献来看，有些问题可处理也可不处理，甚至不处理反而会提高精度，如问题 1）和问题 3）。在进行系统设计时，考虑了一种灵活的方式对待这些问题，即由用户根据分类器在测试集上的精度自主选择对上述问题的处理方式。

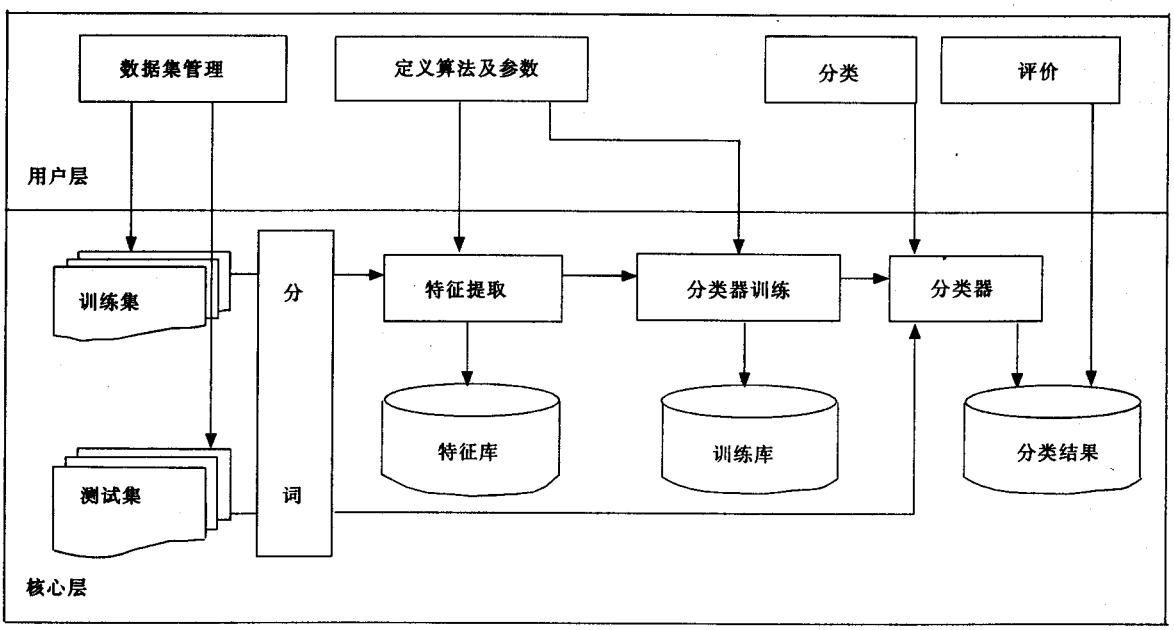

图 1 分类系统体系结构

Fig. 1 Architecture of the classification system

2 基于设计模式的类设计

2.1 分类系统设计要求

对于一个新的分类任务，用户应首先设置好分类类别及该任务对应的运行参数。然后，建立初始的特征表。之后，执行特征子集选取及分类器训练过程，至此，用户就可以应用分类器对新文本进行分类了。在分类器使用过程中，用户可以根据测试结果动态地调整运行参数从而使分类效果最佳。

对一个分类任务而言，它的特征子集、分类器都需要保存下来，以后再使用这些资源时，不需要重新运行特征子集提取及分类器训练的过程。

2.2 类图及其说明

根据系统体系结构及设计要求，设计了图 2 所示的类图。由于篇幅的关系，图 2 中只列出了主要的类及其主要成员，类的说明见表 1。图 2 中涉及到的设计模式[3]如下。1）单例模式：Task 中存放了用户保存的所有分类任务的名字及当前用户打开的任务信息，由于一个分类系统仅有一个 Task 对象，故对它采用了单例模式。2）策略模式：策略模式的用意是针对一组算法，将每一个算法封装到具有共同接口的独立的类中，从而使得它们可以相互替换。由于特征子集选择的算法很多，故对于特征子集的生成，使用了这一模式。Selector 是环境角色；FeatureSelector 充当了抽象策略角色；而 MISelector 等则充当了具体策略的角色。3）模板模式：Task 中的 createClassifier()是一个模板方法，其中内含了在应用一个分类器之前要做的所有步骤，从而使用户仅需要调用 Task 对象的该方法便可以使用需要的分类器对新文本进行分类。这有利于核心层向用户层提供访问接口，而不必关心分类系统内部逻辑。4）迭代子模式：在 CategorySet，FeatureSet，FeatureSubSet 等类中都向外提供了返回一个迭代子的方法，返回的迭代子使人们可以很方便地访问类中聚集的每一个元素，而不必考虑开发人员存储聚集的数据结构，这显然使系统更符合“开一闭”原则（即对扩展开放，对修改封闭）。5）命令模式：这主要是考虑到第 3 节的内容而采用的一个模式，在图 2 中没有体现，将在下面讨论这一问题。

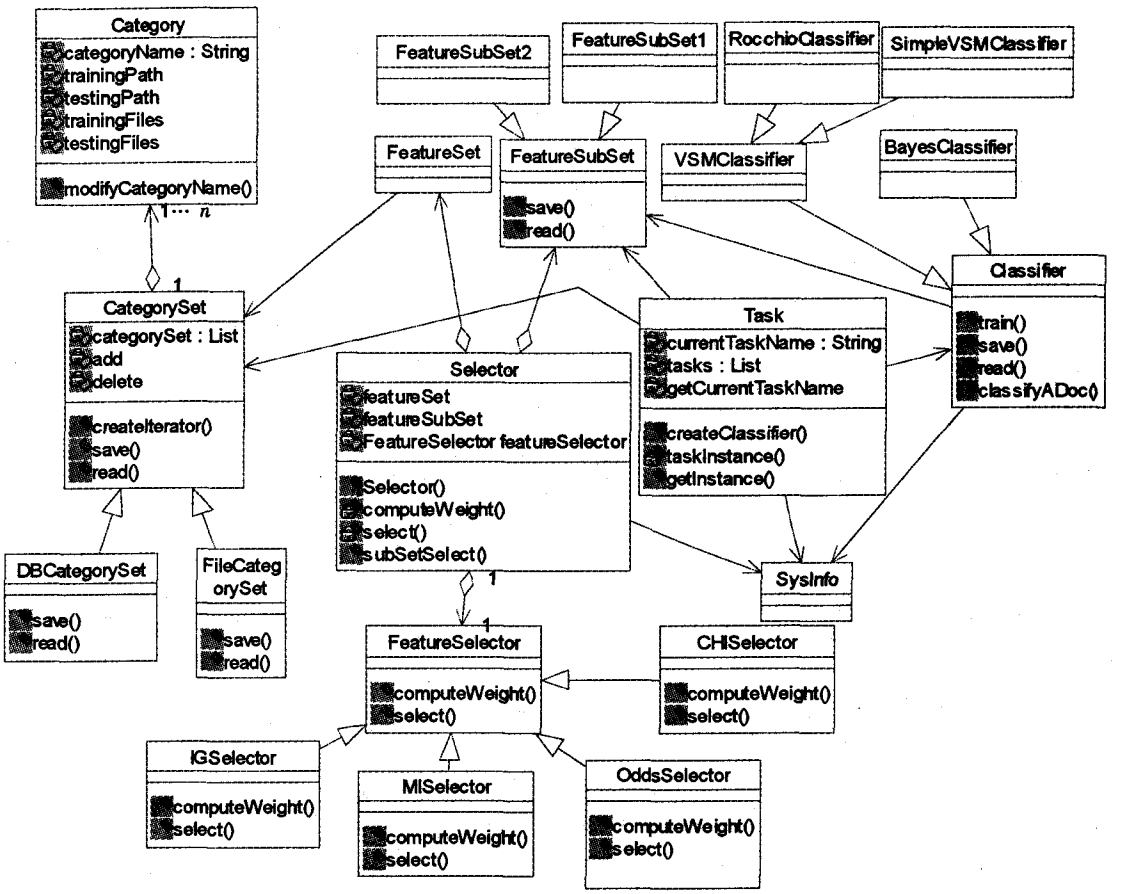

图 2　文本分类系统类图

Fig. 2　Class diagram of the classification system

表 1　类的说明

Tab. 1　Descriptions of the classes

类名	说　明
Task	存放用户保存的所有分类任务的名字及当前用户打开的任务信息
SysInfo	存放运行参数，使用户可以选择对第 2 节描述的基准问题选择处理方式
CategorySet	存放分类类别集及各类别对应的训练集路径、测试例集路径等信息
FeatureSet	存放原始的特征词表
FeatureSubSet	存放特征子集
Classifier	抽象分类器角色，定义各种类型的分类器都要实现的方法
Selector	为特征选择提供方法，在调用相应的选择算法后，向系统返回 FeatureSubSet 对象

3　分类系统的对外接口——“分类语言”模型

为了与其他系统进行无缝连接，分类系统提供了用户层接口。在用户层，用户可以使用分类语言定义一个完整的分类任务，并使用该分类任务中的分类器对新文本进行分类。此外，用户还可以引用核心层对象以生成一个新的分类任务，并使用该任务对应的分类器。第 2 种方式通过直接调用核心层的代码实现。第 1 种方式下，用户通过使用分类语言脚本间接调用核心层功能，这种方式下，不需要了解核心层代码的任何信息。显然，这一种方式更加具有灵活性和可扩展性。然而，这要求人们研究并定义一种分类语言。

笔者提出的分类语言（TCSL）包括 3 类语句：任务定义语句（Task_Def_Statement）、参数定义语句（Para_Def_Statement）及分类语句（Classifier_Apply_Statement）。

〈TCSL〉::=〈Task_Def_Statement〉|〈Para_

Def_Statement〉|〈Classifier_Def_Statement〉

3.1 任务定义语句

〈Task_Def_Statement〉::=〈Task_Statement〉 | 〈CategorySet_Statement〉
〈Task_Statement〉::=new task 〈task_name〉 | open task 〈task_name〉 | delete task 〈task_name〉
〈CategorySet_Statement〉::=delete category 〈category_name〉
| add category 〈categoryName,trainingPath,testingPath〉
| modify category {〈oldCategoryName,newCategoryName〉
| 〈oldTrainingPath,newTrainingPath〉
| 〈oldTestingPath,newTestingPath〉}

3.2 参数定义语句

〈Para_Def_Statement〉::=set stopList 〈true|false〉 | set KNN k=〈value〉
| set featureMethod=〈"IG"|"MI"|"Odds"|"CHI"〉
| set classifier=〈"KNN"|"MNBayes"|"MVBayes"|"Rocchio"|"VSM"〉
| set threshold value=〈value〉 for featureMethod
| set threshold value=〈value〉 for classifier
| set ratio value=〈value〉 for featureMethod

3.3 分类语句

〈Classifier_Def_Statement〉::=〈Feature_Statement〉 | 〈Classifier_Statement〉
| 〈Evaluator_Statement〉
〈Feature_Statement〉::=create feature table | feature subselection
〈Classifier_Statement〉::=classify file 〈filename〉 save 〈resultFileName〉
| classify dir 〈dirName〉 save 〈resultFileName〉 | train classifier
〈 Evaluator_Statement〉::=evaluate file 〈resultFileName〉 save 〈evaluatingFileName〉

从上述定义可见，TCSL 语言模型是一个简单的模型，易于使用者掌握及使用，也易于实现对用户定义的语句进行语法分析。

4 解析、执行语句

用于 TCSL 语句或脚本，系统提供 2 种运行方式，第 1 种是 GUI 运行方式，第 2 种是命令文件解释执行方式。前者通过直接执行系统的可执行程序实现；后者通过在可执行文件后跟上脚本文件名这一参数得以实现，脚本文件内存放用户定义的 TCSL 语句，每一条语句在文件中占一行。下面主要讨论第 2 种方式。

系统首先检查脚本文件的合法性，即其中每一条语句是否符合第 3 节中定义的语句语法。之后检查脚本文件的有效性，即检查脚本文件中的所有命令是否可完成特定的任务。例如：用户定义了一个脚本文件，其中内容如下。

new task test
train classifier

这是一个无效文件，因为用户在新建一个分类任务后，没有定义该任务中包含的分类类别及训练集等信息，此时显然无法对分类器进行训练。一个脚本文件首先应该是合法且有效的，同时，该文件中排在前面的命令不依赖于后面命令的执行结果，则该文件被称为完整脚本文件，否则为不完整脚本文件。显然，一个脚本文件应该是合法的、有效的、完整的才可以得以执行。

脚本中每一条语句的执行结果都会保存在磁盘中，从而使得语句的执行具有“执行命令一次，结果永久使用”的特点。例如，若上一次成功执行了“train classifier”命令，则下次可在成功执行“open task task_name”命令后，直接执行 Classify_Def_Statement 语句而省去“train classifier”这一过程。

为了使系统能解释执行命令，笔者设计了如图 3 所示的类图。

图 3 采用了命令模式，Interpreter 充当客户角色，FeatureSubSet，Task，Classifier 等充当具体命令角色。Interpreter 解释脚本文件中每一条命令，并根据命令生成相应的具体命令角色对象（如果已生成，则不再重复生成），该类角色的 execute() 方法会根据命令中参数设置对象的内部状态，根据命令调用同类角色对象的相应方法，从而生成命令执行结果。

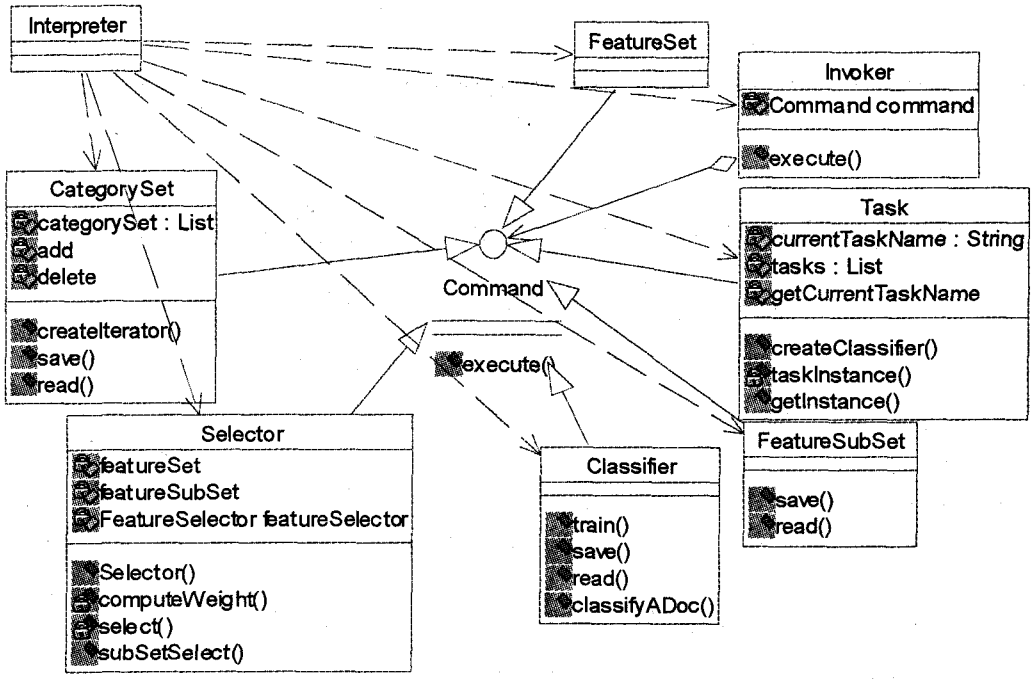

图 3　系统对于语言模型的解释类图

Fig. 3　Class disgram for interpreting commands in TCSL

5　结　语

设计了一个易于扩展及使用的分类系统，对分类中的设计细节进行了较深入的探讨，提出了“分类任务”及“分类语言”模型。该模型有利于分类系统与其他系统的融合。文中提及的设计在作者自行设计开发的文本分类系统得到了初步应用，实践表明此设计是合理的，此工作对开发类似的数据挖掘系统具有一定的借鉴意义。

参考文献：

[1] SEBASTIANI F. Machine learning in automated text categorization[R]. Pisa:Consiglio Nazionale Delle Ricerche,1999.

[2] 李文斌，黄佳进，欧创新，等. 个性化 E-mail 分类器的设计与实现[J]. 北京工业大学学报，2002,28(4):22-26.

[3] 阎　宏. 设计模式[M]. 北京：电子工业出版社，2002.

(上接第 478 页)

参考文献：

[1] BIRD D, MUNOZ C. Automatic generation of random self-cheeking test cases[J]. IBM System J,1983,22(3):229-245.

[2] RAMAMOORTHY C V, HO S, CHEN W. On the automated generation of program test data[J]. IEEE Trans Software Eng,1981(1):117-127.

[3] CYTRON R, FERRANTE J, ROSEN B K, et al. Effieiently computing static single assignnlent form and the control dependence GraPh[J]. Transaetions on Programming Languages and System,1991,13(4):451-490.

[4] GUPTA N, MATHUR A P, SOFFS M L. Automated test data generation using an iterative relaxation method[J]. Software Engineering Notes,1998,23(6):231-244.

[5] 荚　伟，高仲仪. 基于遗传算法的软件结构测试数据生成技术研究[J]. 北京航空航天大学学报，1997,23 (1):36-40.

[6] 傅　博. 基于模拟退火遗传算法的软件测试数据自动生成[J]. 计算机工程与应用，2005,41(12):82-84.

[7] 傅　博. 基于蚁群算法的软件测试数据自动生成[J]. 计算机工程与应用，2007，43(12):97-99.

[8] 李爱国，张艳丽. 基于 PSO 的软件结构测试数据自动生成方法[J]. 计算机工程，2008,34(6):93-94.

[9] 夏　芸，刘　锋. 基于免疫遗传算法的软件测试数据自动生成[J]. 计算机应用，2008,28(3):723-725.

[10] STORE R, PRICE K. Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces[J]. Technical Report, International Computer Science Institute, 1995(8): 22-25.

[11] 黄小城，王希武，常东升，等. 改进的差分演化算法在测试数据生成中的应用[J]. 计算机应用，2009,29(6):1 722-1 724.

[12] 钟治平，徐拾义. 程序插装技术在软件内建自测试中的应用[J]. 计算机工程与应用，2004，40(17):117-118.

[13] 孙冒爱，金茂忠. 基于程序插装的动态测试技术实现[J]. 小型微型计算机系统，2001,22(12):1 475-1 479.

[14] KOREL B. Automated software test data generation [J]. IEEE Transactions on Software Engineering, 1990,16(8) : 870-879.

[15] 姜立强，邱迎锋，刘光斌，等. 利用改进微分进化算法实现线性系统逼近[J]. 电光与控制，2008,15(5):35-37.

[16] 刘明广. 改进差异演化算法求解约束优化问题[J]. 计算机工程与应用，2008，44(19):43-45.