软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(6):1389−1404 [doi: 10.13328/j.cnki.jos.005224]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

一种面向软件配置管理制品的层次分类方法*

徐培兴[1,2], 陈 伟[1], 吴国全[1,2,3], 高楚舒[1], 魏 峻[1,2,3]

[1](中国科学院 软件研究所,北京 100190)

[2](中国科学院大学,北京 100049)

[3](计算机科学国家重点实验室(中国科学院 软件研究所),北京 100190)

通讯作者: 陈伟, E-mail: wchen@otcaix.iscas.ac.cn

摘 要: 配置管理工具(configuration management tool,简称 CMT)作为运维自动化的组成部分,是实现开发运维一体化(development and operations,简称 DevOps)的重要支撑技术.当前,互联网开源社区中存在数量众多的 CMT 脚本制品,但是缺乏有效的层次分类管理,给快速检索和高效利用 CMT 脚本制品带来困难.针对该问题,提出一种面向 CMT 制品的基于在线非结构化描述文档分析的层次分类方法.该方法利用标签共现性关系(tag co-occurrence)建立层次类别体系,基于描述属性特征,实现对 CMT 制品的层次分类器;并使用混合的样本划分方式针对数据倾斜问题进行了改进.对超过 11 000 例训练数据和 1 000 例测试数据进行实验,结果表明:改进的样本划分方式得到的最佳查准率、查全率、调和平均值分别达到 0.81、0.88、0.85,较传统方式查全率提高 0.15,调和平均值提高 0.06.该结果验证了层次分类方法的有效性.

关键词: CMT 制品;层次分类;开源社区;开发运维一体化(DevOps)

中图法分类号: TP311

中文引用格式: 徐培兴,陈伟,吴国全,高楚舒,魏峻.一种面向软件配置管理制品的层次分类方法.软件学报,2017,28(6):1389−1404. http://www.jos.org.cn/1000-9825/5224.htm

英文引用格式: Xu PX, Chen W, Wu GQ, Gao CS, Wei J. Hierarchical categorization for artifacts of configuration management tool. Ruan Jian Xue Bao/Journal of Software, 2017,28(6):1389−1404 (in Chinese). http://www.jos.org.cn/1000-9825/5224.htm

Hierarchical Categorization for Artifacts of Configuration Management Tool

XU Pei-Xing[1,2], CHEN Wei[1], WU Guo-Quan[1,2,3], GAO Chu-Shu[1], WEI Jun[1,2,3]

[1](Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[2](University of Chinese Academy of Sciences, Beijing 100049, China)

[3](State Key Laboratory of Computer Science (Institute of Software, The Chinese Academy of Sciences), Beijing 100190, China)

Abstract: Configuration management tool (CMT), as an essential part of automated system operations, is an important technique to achieve DevOps (development and operations). There are a large amount of reusable CMT artifacts in the internet-scale open source communities and repositories. However, the lack of effective hierarchical categorization leads to the difficulties of effective retrieval and usage of those artifacts. This paper addresses the issue by proposing a hierarchical categorization method for CMT artifacts based on their online unstructured descriptions. This method firstly constructs a category system based on the co-occurrences of tags, and then designs the classifiers based on the features of CMT artifacts, including name and description. To improve the effectiveness of classifications affected by the unbalanced data set, the method builds a hybrid model to divide the sample data. Finally, extensive experiments are carried

* 基金项目: 国家自然科学基金(61402453); 国家重点研发计划(2016YFB1000803)

Foundation item: National Natural Science Foundation of China (61402453); National key Research and Development Plan (2016YFB1000803)

收稿时间: 2016-07-28; 修改时间: 2016-10-11; 采用时间: 2016-12-22; jos 在线出版时间: 2017-02-20

CNKI 网络优先出版: 2017-02-20 15:14:48, http://www.cnki.net/kcms/detail/11.2560.TP.20170220.1514.033.html

out to evaluate the method on more than 11000 CMT artifacts. The results show that this improved method based on hybrid model achieves up to 0.81 precision, 0.88 recall and 0.85 *F*-measure. Comparing to traditional approaches, the recall and *F*-measure of CMT artifacts classification improve significantly. The effectiveness of this method is verified.

Key words: CMT artifact; hierarchical categorization; open source community; development and operations (DevOps)

配置管理工具(configuration management tool,简称 CMT)是实现开发运维一体化(development and operations,简称 DevOps)的重要支撑工具,当前主流的 CMT 采用代码即基础设施(infrastructure as code,简称 IaC)[1]的方式,通过代码来描述目标系统配置,实现自动化的系统安装和配置,从而满足 DevOps 所倡导的持续交付、快速部署和高效运维.基于此,使用 CMT 已成为运维管理领域的主流趋势.据 RightScale 云计算年度报告[2],2016 年,超过 70%的企业正在使用 Chef,Puppet,Ansible 等 CMT 工具.CMT 制品是 CMT 工具用以安装、配置和管理特定的软件系统的可复用执行脚本.在开源技术社区快速发展的背景下,CMT 社区同样积累了大量并且持续快速增长的 CMT 制品,例如,Chef[3],Puppet[4],Ansible[5]中 3 个 CMT 社区已有 4 200 多用户,贡献超过 14 000 个 CMT 制品,其中,Ansible 社区自 2014 年至今,其 CMT 制品数量由 660 增长为 6 895,两年间增长达 10 倍,而且正吸引越来越多的贡献者.另一方面,当前软件制品社区如 OpenHub[6],SourceForge[7]等已有超过 670 000 个软件,在 DevOps 的发展趋势下,如果能全面实现自动化部署和配置,那么 CMT 制品将出现更大规模的增长.

但是,大量存在的 CMT 制品在提升工具的易用性和制品的复用性的同时,也给正确选择和使用 CMT 制品带来了困难.当前主流 CMT(如 Puppet,Ansible)的官方社区仅提供简单资源列表以及关键字匹配的搜索服务,用户在查找与需求匹配的脚本制品时,需要花费大量时间对搜索结果进行浏览和细化.例如,用户在 Puppet 社区中以 monitoring 作为关键字搜索面向监控软件的 CMT 制品时,得到包括诸如 zabbix[8],nagios[9]等在内的多达 124 个脚本制品,用户仍需要额外花费大量的精力和时间从中找到满足特定系统和版本等需求的脚本制品.其他脚本制品库如 Ansible,Chef 社区等也存在类似的情况.针对上述问题,部分资源库采用标签(tag)的方式来增强检索的准确性,但标签可由开发者自定义创建,缺乏规范性,极大依赖于制品开发人员的领域知识和标签使用习惯.其次,标签的扁平化特征无法反映出标签之间实际存在的层次化关系.因此,软件运维管理领域迫切需要一种 CMT 制品的自动化层次分类机制,以满足用户快速检索和准确定位目标制品的需求.层次化分类将大规模数据按照特征逐级划分,细化范围,将层次分类应用于 CMT 制品,不仅提高检索效率和准确度,还能够构建脚本制品间层次关联关系,有助于脚本制品的维护管理,提高 CMT 制品的利用率.但是,CMT 制品的层次分类面临以下难题.

1) 与传统的软件库(如 source forge)不同,当前 CMT 领域并不存在面向 CMT 制品的可用的、预定义的层次分类体系,因而也没有已进行层次类别标注的实验数据;即使部分 CMT 制品库采用标签机制进行标识,但是无法反映标签间的层次化关系;
2) CMT 制品与特定的领域编程语言相关[10].例如,Chef,Ansible 和 CFEngine[11]使用命令式(imperative)的脚本语言实现面向特定软件系统的部署配置操作自动化,而 Puppet 和 SmartFrog[12]]则通过声明式(declarative)语言描述期望达到的目标状态来实现任务的自动执行.因此,无法使用传统的基于程序分析的方法[13−15]来实现制品分类.

针对上述问题,本文提出了一种基于制品描述信息(profile)的 CMT 制品层次分类方法:首先,该方法以 CMT 制品库中频繁使用的标签作为类别标识,通过挖掘这些标签间的隶属关系来构建层次化分类体系;然后,该方法基于监督学习训练并建立一组分类器支持 CMT 制品的自动分类.方法通过标签匹配(tag matching)对 Puppet 和 Ansible 的 CMT 制品进行标注并将其作为训练数据,接着以 CMT 制品描述信息中的名称(name)和描述(description)作为描述文档,采用 TF-IDF(term frequency-inverse document frequency)[16]抽取文档特征,最后使用机器学习方法实现 CMT 制品的分类器.

最后,本文基于对 3 个主流 CMT 资源库超过 11 000 个制品进行层次化分类实验进行评价.实验结果表明:(1) 本文方法抽取了 90 个细粒度的类别标识,并构建了基于这些类别的 4 层分类体系,有助于用户更加准确的定位目标 CMT 制品;(2) 本文方法在 CMT 制品分类方面具有较好的准确率(precision)、召回率(recall)和调和

平均数(*F*-Measure),分别达到 0.81,0.88 和 0.85.

本文的主要贡献归纳如下:

1) 提出一种面向 CMT 制品的层次分类方法.该方法基于制品描述信息提取分类特征,不依赖于特定领域语言,能够对制品实施细粒度的层次化分类;
2) 本文面向超过 11 000 个 CMT 制品构建了包含 90 个分类标识、细粒度的多层次分类体系;
3) 基于多个 CMT 社区制品数据的实验和方法评价.本文选取当前主流 CMT 工具的上万个脚本制品及其社区数据为对象,对文中提出的分类方法进行了实验、对比和评价.实验结果表明:本文方法在调和平均值方面达到 0.85,较类似的软件制品分类方法[17]提高 0.2;使用混合方式的正反面样本划分策略改进分类模型,相比传统划分策略,查全率提高 0.15,调和平均值提高 0.06,整体分类效果得到有效的提升.

本文第 1 节介绍研究动机,并在此基础上对本文方法进行概述.第 2 节详细介绍面向 CMT 制品的层次分类方法.第 3 节基于本文方法开展 CMT 制品分类的相关实验和结果评价分析.第 4 节介绍相关研究工作.最后,第 5 节对本文工作进行总结.

1 研究动机

配置管理工具通过 CMT 制品对应用进行部署和配置管理,CMT 制品由特定领域语言(DSL)实现,一般包含配置代码、安装文件、模板、说明文件等部分,是独立的可复用和共享的配置单元.例如:cookbook[18]是 Chef 进行配置和管理的脚本制品,每个 cookbook 包含 recipes(配置代码)、参数、安装文件、模板、说明文件;Puppet 制品包含 manifests(配置代码)、模板、说明文件等.

制品由各自的 CMT 社区知识库(如 puppet forge,chef cookbook supermarket[3])管理,开发者向知识库中提交自己开发的制品时,要提供制品的名称、描述、标签以方便识别.图 1 展示了名称为“BoxUpp/mysql”用以安装配置 MySQL 的 Puppet 制品主页,显示了制品的名称、描述、标签等属性.

Fig.1 Page of BoxUpp/mysql in Puppet Forge

图 1 Puppet Forge 中的 BoxUpp/mysql 主页

目前,Puppet,Ansible 和 Chef 是最流行的配置管理工具,3 种工具的知识库共有超过 14 000 例 CMT 制品.表 1 列出了不同知识库中的 CMT 制品示例.这些制品功能相似但名称和描述不完全相同,其中,Chef 知识库中的制品没有标签属性.

Table 1 Examples of CMT scripts

表 1 CMT 制品示例

资源库	名称	标签	描述
Puppet	boxupp/mysql	mysql, database, db, mysqlserver	A puppet module for installing and configuring mysql
	puppetlabs/mongodb	mongodb, mongo, database, cluster, nosql, sharding	Installs MongoDB on RHEL/ Ubuntu/Debian.
Ansible	micropyramid.mysql	database, sql, mysql, Web	Ansible script to install MySQL over Ubuntu server.
Chef	mysql	–	Provides mysql_service, mysql_config, and mysql_client resources

为了将上述 CMT 制品的进行自动化层次分类,需要解决以下技术问题.

1) 不同于文献[17]关于软件制品层次分类的已有工作,CMT 制品没有预定义的层次类别,如何进行划分?

2) 标签能够在一定程度上标识制品类别特征,如何从扁平化的标签中构建出层次类别?

3) 对于某个层次类别,如何建立与具体 CMT 制品描述信息(如名称、描述、标签)的关联?

4) 不同领域的制品数量相差很大,如 Ansible 知识库中有 3 000 例制品标注“system”标签,而被标注为“server”的只有 45 例,如何处理类似非平衡的数据集?

通常情况下,开发者为所提交制品进行标注时会选择与制品本身功能特性符合的关键字,如表 1 中的标签 database,sql,db,mysql 等,类似标签在 MySQL 相关的制品中大量出现.因此,可以将标签作为制品的类别主题.同时,通过观察发现,CMT 制品的不同标签间隐含了隶属关系,例如,BoxUpp/mysql 与 puppetlabs/mongodb 含有相同的标签 database,这表明,database 比 mysql 和 mongodb 更一般化,即,mysql 和 mongodb 可以作为 database 的子类别(subcategory).基于此,能够建立标签层级分类,并将带有标签属性的制品进行划分.

以表 1 中所列出的制品为例,CMT 制品的名称反映该制品所管理的应用,描述简要介绍了制品功能,标签以关键字形式概括该制品的特征.同时,一些相同关键词频繁出现在相似的制品描述中,如表中有 3 个 CMT 制品描述都出现了 mysql,该词也出现在其标签列表中.这表明 CMT 制品的名称、描述与其标签具有关联性,注意到 Chef 制品没有标签属性,因此可以利用 CMT 制品共有的名称和描述作为特征,对 Chef 制品进行聚合并划分到已有的分类体系中.

我们依据标签,将 Puppet 和 Ansible 的制品匹配到对应类别中作为训练集,利用名称和描述的文本特征进行分类器训练,以实现对无标签 Chef 制品的层次分类.为了应对非平衡的分类数据集问题,本文提出一种混合策略,依据分类树中兄弟类别数目,在同层次类别划分中分别使用 one-against-rest 和 divide-by-2[19]的两种样本划分方法.

2 CMT 制品层次分类方法

图 2 展示了面向 CMT 制品的自动层次分类过程,该过程主要有 3 个步骤.

1) 构建层次类别.从 Puppet 和 Ansible 知识库中爬取包含标签属性的 10 790 个 CMT 制品,找出常用标签作为类别关键字,从中挖掘出隶属关系来构建层次分类;

2) 训练分类器.使用标签匹配方法将 Puppet 和 Ansible 的 CMT 制品标注到对应分类中,再基于有监督学习方法,通过训练上述 CMT 制品的说明文档特征(名称、描述)构建一系列层次化的分类器;

3) CMT 制品分类.使用得到的分类器实现对其他 CMT 知识库(如 Chef)的制品的层次分类.

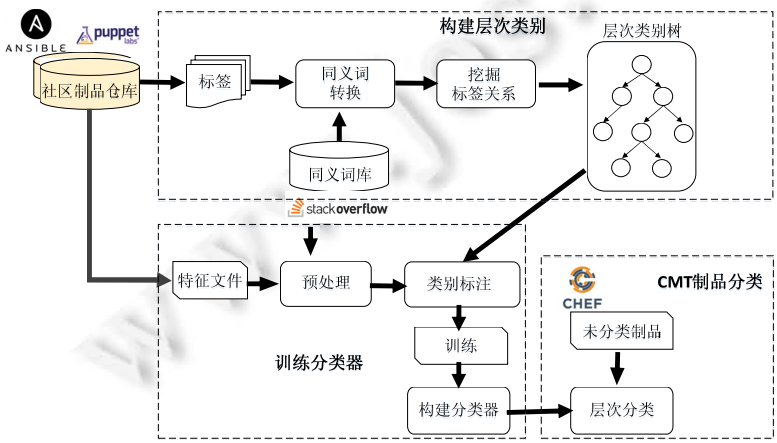

Fig.2 Process of classification for CMT artifacts

图 2 脚本制品层次分类过程

2.1 构建层次类别

部分软件社区存在软件制品的分类体系,如 SourceForge 社区将软件分为 Audio&Video,Business& Enterprise 等 10 类别,每个类别内再进行更细粒度的划分.CMT 制品的分类与软件分类相似,但 CMT 制品本身用于管理和配置软件,尤其是复杂的大规模软件系统,如数据库系统、负载均衡、Web 应用等,这些系统大部分运行于类 Unix 平台,以开发类、工具类为主,与通用软件的领域有所不同,因此,目前的软件制品分类体系无法满足 CMT 制品的分类需求,有必要构建 CMT 领域的分类层次.

正如前文所提到,标签能够反映制品的类别主题,因此,我们使用标签作为类别关键字构建分类.然而标签由开发者自行选择关键字标注,受限于开发者领域知识差别,存在一些同义词标签,因此需要将标签进行标准化.我们使用 StackOverFlow 社区的同义词库[20]来实现这一过程,如,将 mongo,mongod 转换为标准描述 mongodb,将 zmq,0mq 转换为标准描述 zeromq.同时,为保证作为类别关键字的标签具有代表性,排除拼写错误,我们过滤掉出现频度小于阈值(30 次)的标签.

目前已有标签是扁平化的,如表 1 中,mysql,sql,database 等标签共同标注 MySQL 制品,不能直接反映标签间的逻辑隶属关系;然而,database 标签还会出现在 mongodb 制品中,我们使用标签对(mysql,database)及(mongodb,database)分别表示 database 标签在两个制品的出现.database 同时概括两个制品的特征,表明该标签可以作为 mysql,mongodb 逻辑上的父类标签,因此,我们基于该标签共现(tag co-occurrence)[21]在大量标签对中挖掘隶属关系.

对于给定的两个标签 A 和 B,我们使用 $A\subset B$ 表示 A 是 B 的子类别.如果满足如下条件,我们认为 $A\subset B$ 关系成立.

1) 如果某一制品被标注 A 标签,那么该制品很可能被同时标注 B 标签.即:A 标签出现情况下,其父标签 B 很可能同时出现.我们使用 $A\rightarrow B$ 表示这一可能性;
2) $|A|<|B|$.即:对于所有的制品,带有 A 标签的制品数量小于 B 标签的数量.

我们使用 $p(t)$表示标签 t 出现的可能性:

$$p(t)=\frac{|t|}{N} \tag{1}$$

其中,N 为 CMT 制品的总数,$|t|$是所有制品中带有标签 t 的数量.

基于上述条件,标签共现 $A\rightarrow B$ 可使用公式(2)表示:

$$\begin{aligned} &support(A\rightarrow B)=p(A\cap B)=\frac{|A\cap B|}{N}\\ &confidence(A\rightarrow B)=p(B\mid A)=\frac{p(A\cap B)}{p(A)}\\ &lift(A\rightarrow B)=\frac{p(B\mid A)}{p(B)} \end{aligned} \tag{2}$$

其中,共现支持度 $support(A\rightarrow B)$表示同时标注 A 和 B 标签的制品比例;共现置信度 $confidence(A\rightarrow B)$表示当制品被标注 A 后,B 标签出现的后验概率;提升度 $lift(A\rightarrow B)$表示 A,B 标签相关性的验证.

当上述 3 个值满足公式(3)时,可认为 $A\rightarrow B$ 的隶属关系成立:

$$support(A\rightarrow B)>\alpha_s, confidence(A\rightarrow B)>\alpha_c, lift(A\rightarrow B)>\alpha_l \tag{3}$$

其中,$\alpha_s,\alpha_c,\alpha_l$ 分别表示最小支持度、最小置信度以及提升度阈值.

通过上述分析,我们从 10 790 例 Puppet 和 Ansible 制品中得到了带有隶属关系的标签对集.表 2 所示 nosql,mongodb,redis,elasticsearch 是 4 个常用标签,一方面,其他标签都与 nosql 频繁同时出现;另一方面,nosql 的频度要多于其余 3 个标签,因此可以推断出,nosql 可以作为 mongodb,redis,elasticsearch 的父类别.

同时需要注意到,隶属关系具有传递性.例如,mysql,sql,database 三者中,$mysql\subset sql$ 且 $sql\subset database$,则有 $mysql\subset database$.因此,可以通过树形结构表示层次分类,向树中添加新的带有隶属关系的标签对,最终实现整个分类树的构建.算法 1 展示了层次分类树的构建过程.该算法以宽度优先搜索构建层次分类树.首先,使用一个

Root 节点初始化分类树(M_{tree}),遍历标签集(T),找到所有无父类别的顶级标签,将其作为 *Root* 的孩子节点并添加到节点标签队列(q_t)中.然后,对于节点队列中的每个标签(t),从该标签相关的标签对集合(Sub_t)中找到子标签集(T_c),检测该集合中的标签(t_c)是否已存在于分类树中:如果不存在,直接将(t,t_c)作为新的分支添加到树中,并将 t_c 加入队列 q_t;否则,比较已有标签对($t_c.parent$,t_c)与(t,t_c)的置信度,保留置信度高的分支.最后得到以分类树形式表示的层次类别.

Table 2 Example of extracting hierarchical relations

表 2 隶属关系示例

标签	频度*	共现频度**
nosql	214	–
mongodb	95	51
redis	69	40
elasticsearch	105	43

*频度指标签标注的制品数量 **共现频度指标签与“nosql”共同标注的制品数量

算法 1. 构建层次分类树算法.

Input:标签集 T,标签对集 Sub_t,分类树根节点 *Root*;

Output:层次分类树 M_{tree}.

```
q_t=∅;
M_tree=Root;
for (i=0; i<T.size(); i++) do
   t=T.get(i);
   if (t.parent()==null) then
      q_t.add(t);
      Root.appendChild(t);
for (j=0; j<q_t.lenght(); j++) do
      t=q_t.get(j);
      T_c=Sub_t.getChildren(t);
      /*get subcategory tags of t*/
      for each t_c in T_c do
         if (q_t.contian(t_c)) then
            if (confidence(t_c.parent,t_c)<confidence(t,t_c)) then
               /*keep the tag pair with max confidence*/
               T_c.parent.removeChidl(t_c);
               t.appendChild(t_c);
            else
               q_t.add(t_c);
               t.appendChild(t_c);
   return M_tree;
```

2.2 训练分类器

我们使用 Puppet 和 Ansible 知识库中的制品作为训练数据集,把说明文档作为分类特征,采用基于二元分类器的自上而下(top-down)分类方法[22],并应用监督学习的几种分类算法来训练分类模型.

CMT 制品的说明文档是制品名称与描述特征的文本聚合,我们采用应用最广泛的 TF-IDF 模型[15]实现文本特征向量的抽取.该模型基本思想是:如果词 w 在某类文档中出现频率高,而在其他文档中很少出现,则词 w 具

有很好的类别区分能力.公式(4)展示了词项 w 在文档 d 中的 TF-IDF 的计算过程:

$$\left.\begin{aligned} TF_{w,d} &= \frac{n_{w,d}}{\sum_k n_{k,d}} \\ IDF_w &= \log\frac{|U|}{n_w} \\ TF\_IDF_{w,d} &= TF_{w,d} \times IDF_w \end{aligned}\right\} \tag{4}$$

其中,$n_{w,d}$ 表示词 w 在文档 d 出现的频率,n_w 表示含有词 w 的文档个数,$|U|$表示文档全集的个数.

针对多类别层次分类,有两类主要的分类方法:全局层次分类(big-bang)和自上而下分类(top-down)[22].全局层次分类对同层次整个类别学习一个多类别的分类器,然而这种方法在类别总数较多时难以训练出效果良好的统一分类器.相反,自上而下分类方法采用分而治之的策略,按照类别层次自上而下对每一个类别单独构建分类器,分解为局部分类问题,然后在某类别内部再次构建分类器.文献[23,24]指出,自上而下分类方法在时间、空间开销方面都要优于全局层次分类.因此,本文采用自上而下的分类方法.

在分类算法的方面,目前有支持向量机(SVM)、k 最近邻(kNN)以及朴素贝叶斯(Naïve Bayes)等多种算法,其中,SVM 算法是文本分类的最有效方法之一[25,26].因此,本文将使用 SVM 算法实现层次分类器;同时,也分别基于 kNN 及 NB 算法进行实验以验证和对比分类效果.

上一节我们从标签中得到了层次类别,在这里,我们使用标签匹配的方式,将带有标签的制品分别对应到不同层次的类别中.在构建分类器时,我们为每个类别都训练一个二类分类器,这就需要确定该分类器的正面和反面样本.由于自上而下的分类方法使用分而治之策略,因此,这里的正反面样本都是在同一父类别范围内而言.

在多类别的分类问题中,正反面样本的划分方式主要有 3 种[19].

1) one-against-one 方法.每两个类别的数据分别作为正反面样本构建一个二类分类器,n 个类别将有 $n(n-1)/2$ 个分类器,新样本的预测取决于分类器的分类选举;这种方法分类器规模庞大且复杂;
2) one-against-rest 方法.每个类别的样本与剩余所有类别的样本分别作为正反面样本,n 个类别将构建 n 个二类分类器,新样本通过 n 个二类分类器预测;
3) divide-by-2(DB2)方法.将类别划分成样本规模相当的两个组,分别作为正反面样本,构建这两个组的分类器;然后,每个组内继续二分.如图 3 所示,以类别数目为 5 的分类问题为例,首先将 5 个类划分成两组(a,b,c)及(d,e),再分别进行组内划分,最终构建了 4 个分类器来区分 5 个类别.

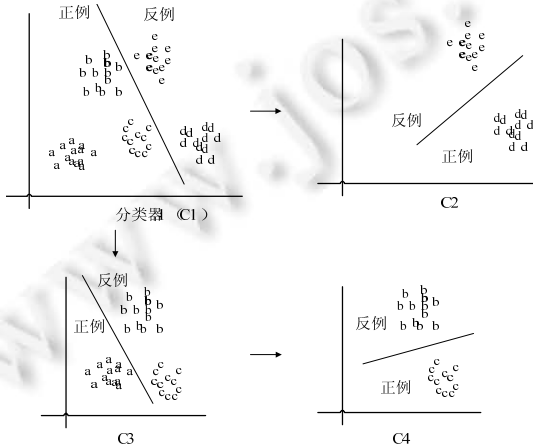

Fig.3 Diagram of two-divide model

图 3 divide-by-2 模型示意图

在 CMT 制品样本中,不同类别的样本数量相差较大,分布不均衡,这将导致数据倾斜(unbalanced training

data)问题.例如:在 Ansible 制品中,标注为 system 标签的制品数量超过 3 000 例,而标注为 server 的仅有 45 例;而在同层级类别较多时,使用 one-against-rest 划分样本,某类别样本与该层级所有其他类别的样本也将形成数据倾斜.不平衡的数据集严重影响分类准确度,为应对该问题,本文提出一种混合的正反面样本划分策略,将同一父类别的兄弟类别数目作为决策标准,当该数目超过设定阈值($n_{sibling}$)时,使用 divide-by-2 方法划分样本为两组,直到组内类别数目不超过 $n_{sibling}$;否则,使用 one-against-rest 方法.其中,divide-by-2 方法能够减小数据倾斜,控制分类模型的复杂度, one-against-rest 方法减小分类的出错率.

2.3 分类预测

在训练得到分类器后,我们使用逐步细化的方法对 CMT 制品预测分类.算法 2 展示了对于给定新制品样本的分类过程.

算法 2. 分类预测算法.

```
Function main(Root,m)
  Input:分类树根节点 Root,待分类制品 m;
  Output:目标类别 t_m.
  return findCategory(Root,m);
Function findCategory(t,m)
  Input:类别节点 t,待分类制品 m;
  Output:目标类别 t_m.
  t_cur=t;
  t_m=t;
  q_children=getChildren(t_cur);
  /*get subcategories*/
  If (q_children.isEmpty()==false) then
    isFind=fasle;
    /*a variable shows ending of search*/
    for (i=0; i<q_children.length(); i++) do
      t_sub=q_children.get(i);
      If (belongsTo(m,t_sub) then
        isFind=true;
        return findCategory(t_sub,m);
        /*execute recursively*/
    If (isFind==false) then
      return t_m;
  else
    return t_m;
Function belongsTo(m,t)
  Input:类别 t,待分类制品 m;
  Output:如果 m 属于 t,输出真;否则为假.
  t_classifier=t.getClassifier();
  /*get the trained classifier*/
  m_feature=tf-idf(m);
  /*get the TF-IDF feature vector of m*/
    If (1==t_classifier.predict(m_feature) then
```

```
        return true;
    else
        return false;
```

算法 2 中,*belongsTo* 方法判断制品样本 *m* 是否属于特定的类别 *t*,该方法从 *m* 中解析名称和描述作为说明文档,并计算出该文档的 TF-IDF 特征向量($m_{feature}$),然后将该向量作为分类器预测的依据.我们从层次分类树的根节点 *Root* 开始,调用 *findCategory* 方法,以递归形式搜索 *m* 的目标类别,如果给定制品样本 *m* 属于当前节点类别 *t*,那么 *m* 将继续在 *t* 的 *n* 个子类别($t_1,t_2,\ldots,t_n$)中继续划分,直到 *m* 不属于 *t* 的任何子类别或搜索到达叶子节点.对于第 1 种情况,当前的节点类别 *t* 即是制品的目标类别;第 2 种情况下,所到达的叶子节点类别是目标类别.考虑到某个制品可能属于多个分类的情况,该算法对制品在多个类别中分别预测,因此无需特别改动.

图 4 展示了 database 类别下 sql,nosql 等子类以及其包含的 CMT 制品情况.算法 2 将 puppetlabs/mysql 以及 arioch/redis 分别划分到 mysql 和 redis 类别下,然而 golja/influxdb 在被划分到 database 类别后没有进一步找到子类别,因此将其直接划分到 database 类别下.

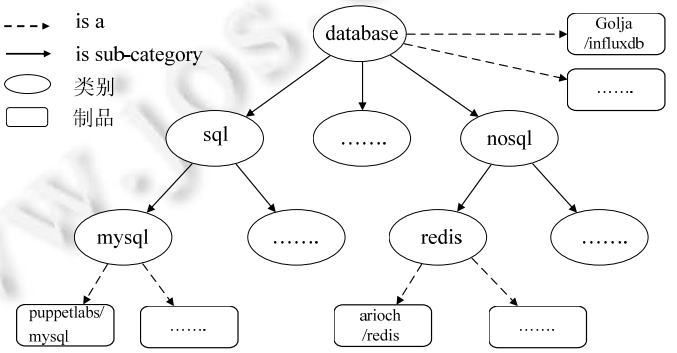

Fig.4 Node of database and its children in hierarchical category tree

图 4 层次分类树中 database 节点示意图

3 实验及结果分析

本文设计了 3 组实验,分别验证和评价本文的分类树构建方法和 CMT 制品分类方法的有效性,并评估混合方式正反面样本划分策略对分类效果的影响.

3.1 实验数据预处理

本文基于 Python 的 urlib2[27]和 Beautiful Soup[28]实现了爬虫系统,用以自动地从 Puppet,Chef 和 Ansible 制品库中爬取 CMT 制品.对于每个制品,本文方法进行如下预处理.

1) 解析元数据中的说明文件,抽取制品名称、描述、标签,去除无描述和标签的制品数据;
2) 对 3 个属性进行分词处理,利用停用词库去除 a,of 等停用词,去除&,*等无义字符,保留词干;另外,数据中存在同一词语有多种表述形式的情况,如 zeromq 可能被拼写为 zmq 或 0mq,论文从 Stack Overflow[20]社区爬取了同义词库,包含 2 900 组同义词,通过同义词替换,将相同含义的不同词汇规范成统一表述;
3) 将 CMT 制品名称和描述聚合,作为该 CMT 制品描述文档,而标签用于构建层次分类树.

经过上述处理,得到数据集详细信息见表 3,最终得到来自 Puppet 和 Ansible 社区的 10 790 例有效数据.除此之外,我们还爬取了 2 720 个 Chef 脚本制品.由于 Chef 制品没有分类标签,我们对其中的 1 000 例进行人工分类标注,作为后续实验验证部分评价方法有效性的基准(baseline).

Table 3 Details of dataset

表 3 数据集详细信息

资源库	脚本制品数量	平均描述词语个数	平均标签个数	是否可用于训练
Puppet	3 895	7.69	4.95	是
Ansible	6 895	8.68	4.64	是
Chef	2 720	6.84	–	否

3.2 实验评价指标

分类问题通常采用查准率(precision)、查全率(recall)以及调和平均数 F(F-measure)来评价分类器的效果,对于层次分类问题,需要对所有分类器的整体分类效果做出评价,因此,本文采用微平均的方式综合模型中所有分类器的分类效果,这也是层次分类研究常用到的度量方法之一[29],见公式(5).

$$\left.\begin{aligned} Micro\_P &= \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FP_i} \\ Micro\_R &= \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FN_i} \\ Micro\_F_1 &= \frac{2 \times Micro\_P \times Micro\_R}{Micro\_P + Micro\_R} \end{aligned}\right\} \tag{5}$$

微平均考虑每个分类器样本规模在整体模型中的权重,计算整体分类器模型的查准率、查全率,然后计算调和平均值 F_1.公式(5)中,TP_i,FP_i,FN_i 分别表示类别 i 的真阳性(ture positive)、假阳性(false positive)和假阴性(false negative)样本数量.

3.3 分类树构建实验

本文在第 2.3 节介绍了类别标签隶属关系的挖掘方法,本节采用划分置信度梯度方法进行实验,通过观察有向标签对在不同置信度下隶属关系的合理性,来确定最小置信度α_c.本实验中,设定有向标签对的最小支持度α_s=0.0005,提升度α_l=1,把每 10 000 例样本中标注 5 个以上且具有正向关联关系的共现标签作为候选,共得到 891 个符合 $A \rightarrow B$ 关系的有向标签对.首先把有向标签对按照置信度降序排列,以 0.1 为梯度将所有标签对划分为 10 组;然后随机抽取每组中 20%的标签对,通过人工判断标签对隶属关系是否是合理的(reasonable);最后统计该组的合理比例,得到表 4 的统计结果.

Table 4 Statistic data of tag pairs with different confidences

表 4 不同置信区间的标签对统计

标号	置信区间	有向标签对个数	合理比例
1	(0.9,1.0]	52	1
2	(0.8, 0.9]	32	1
3	(0.7, 0.8]	42	1
4	(0.6, 0.7]	45	0.80
5	(0.5, 0.6]	94	0.70
6	(0.4, 0.5]	100	0.60
7	(0.3, 0.4]	131	0.46
8	(0.2, 0.3]	131	0.38
9	(0.1, 0.2]	157	0.31
10	(0, 0.1]	107	0.36

从表中可看出,标签对的合理比例随置信区间的下移而降低,在区间(0.3,0.4]下降到 0.5 以下.当置信度在(0.4,1.0]区间内时,各组标签对的合理比例都在 0.6 以上,对该区间内各组合理比例加权计算,得到整体合理比例为 0.79.文献[23]中对社会化标签提出一种整合了主题模型的层次聚类方法,平均正确率达到 0.79.参考此基准,本文将最小置信度α_c 设置为 0.4,对于构建分类树是比较好的选择.

我们对置信度在(0.4,1.0]内的标签对应用分类树构建算法,得到了包括 4 层的 90 个类别的分类树体系,其中,第 1 层类别下有包括 system,server 等 9 个分类,第 2 层~第 4 层分别包含 33,42,6 个分类.然后,通过标签关键

字匹配的方法,对所有的 10 790 例 CMT 制品进行类别标注,表 5 列出了类别及标注结果的统计数据.需要注意的是:由于父类别的样本包括了子类别样本,表中不同层次平均样本数存在重复计数.本文另外对获取的 Chef 制品中 1 000 例样本进行了人工标注,该数据在后续实验中作为测试样本用于验证.

Table 5 Statistic data of hierarchical categories

表 5 分类层次统计数据

分类层次	类别个数	兄弟类别数目			正面样本数目		
		最小	最大	平均	最小	最大	平均
1	9	9	9	9	352	3 320	997.67
2	33	1	7	3.7	31	672	208.24
3	42	1	5	2.3	30	317	80.13
4	6	1	2	1.5	36	227	72.8.

3.4 CMT制品分类实验

本实验用以分析评价不同分类算法对于最终分类结果所产生的影响.本实验在仅使用 one-against-rest 方法划分正反面样本情况下,分别采用支持向量机(SVM)、k 最近邻(kNN)、朴素贝叶斯(Naïve Bayes)这 3 种算法作为基本分类器,对比分析 3 种分类算法在 CMT 制品层次分类问题的效果.实验中使用了基于 Python 的 Scikit-Learn[30]机器学习方法库,3 种算法均使用 Scikit-Learn 方法库的默认参数.

实验将 CMT 制品描述文档中词语的 TF-IDF 作为特征向量,首先使用 Puppet 及 Ansible 社区的 10 790 例 CMT 制品进行 5 折交叉验证(cross validation),将数据随机等分为 5 份,使用其中 4 份作为训练,另外 1 份作为测试,循环做 5 次实验,所得到分类器指标为 5 次的平均值.图 5 展示了交叉验证的实验结果.

从图中可看出:3 种算法中,SVM,kNN 与 Naïve Bayes 查准率相差不大,都达到 0.9 左右;3 种算法查全率则要远低于查准率,最大差距在 0.2 以上,尤其是 kNN 算法查全率最低;综合查准率与查全率的调和平均值 F_1 也出现明显差别.总体上,SVM 算法要优于其他两种算法,这个结果与相关研究的结果是一致的.

在上述实验基础上,本文将 10 790 例制品用作训练数据,另外对 1 000 例 Chef 制品人工标注,作为新的实验测试数据,在相同实验环境和参数下再次进行验证,结果如图 6 所示.该实验结果相比于交叉验证实验,3 种算法各项评价指标都有所下降;其中,Naïve Bayes 算法的查全率与调和平均值 F_1 下降明显;而 SVM 算法的性能较稳定,仅有约 0.07 的下降幅度,仍然是三者中最优的分类算法.由于新测试数据和训练数据分别属于不同的 CMT 社区,开发者的领域知识及文档的撰写习惯有所差别,导致文本数据中词语的分布稍有不同,所以出现分类性能的下降.其中,朴素贝叶斯算法对数据形式最为敏感,因而在两次实验中性能指标下降幅度最大.由于第 2 次实验是在实际分类场景下的对新数据的测试,因此本文将该结果作为层次分类器的实际效果.从两次实验结果总体来看,查全率严重影响了整体分类效果.我们分析原因是:实验中仅使用 one-against-rest 的样本划分方法,单一类别的正样本数量与剩余类别的总体负样本数量相差巨大,出现数据倾斜问题,影响了查全率.

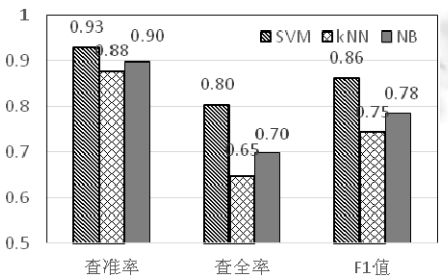

Fig.5 Cross calidation of categorization based on algorithms of SVM, kNN and Naïve Bayes

图 5 分别使用 SVM,kNN,Naïve Bayes 算法的交叉验证实验结果

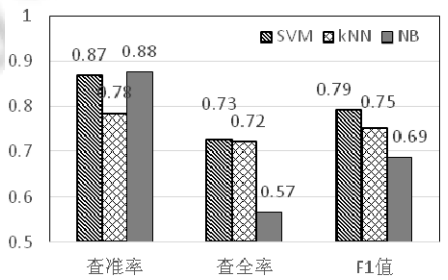

Fig.6 Result of categorization using new test dataset of Chef

图 6 使用 Chef 新测试数据集的实验结果

3.5 基于SVM的混合样本划分实验

上组实验中,SVM 算法取得了最好的效果.本节基于 SVM 算法,在相同实验环境下,采用 one-against-rest 与 divide-by-2 结合的混合样本划分模型进行实验.当分类树中同层次兄弟类别数目大于类别阈值时,使用 divide-by-2 划分正反面样本,否则使用 one-against-rest 确定正反面样本.每次 divide-by-2 划分都将样本数据分为规模相当的两组,直到组内类别数小于阈值为止.该类别阈值能够表示 divide-by-2 方法在模型中的参与程度,由于分类树中最大兄弟类别数目为 9,因此本文调整该阈值大小,在[2,10]区间内进行多次实验,以验证混合样本划分模型的有效性.最终的实验结果如图 7 所示.

从图 7 可以看出:随着兄弟类别阈值降低,查准率从 0.91 下降到 0.77;查全率从 0.69 提升到 0.92;两者的调和平均值 F_1 从 0.78 上升到 0.85 后稍有下降.当类别阈值为 4 时,调和平均值 F_1 达到最大,分类效果最好.当兄弟类别阈值为 10 时,查准率、查全率以及调和 F_1 值退化与上组实验结果接近.

图 8 将类别阈值为 4 时的混合样本划分实验结果与仅使用 one-against-rest 原始划分方法的实验结果对比.从图中可以看出:相比于原始划分方法,混合样本划分方法准确度稍有下降,但查全率大幅提升,整体调和平均值 F_1 比原来提升 0.06.这表明混合样本划分方法能够有效减小数据倾斜,提升层次分类效果.

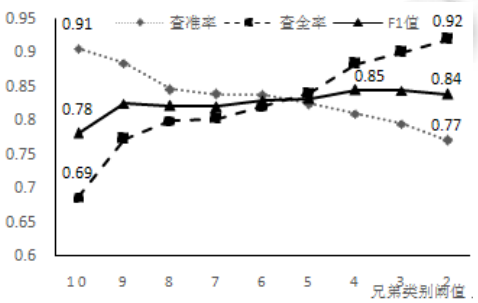

Fig.7 The impact of threshold of brother-class-number

图 7 不同兄弟类别阈值对分类结果的影响

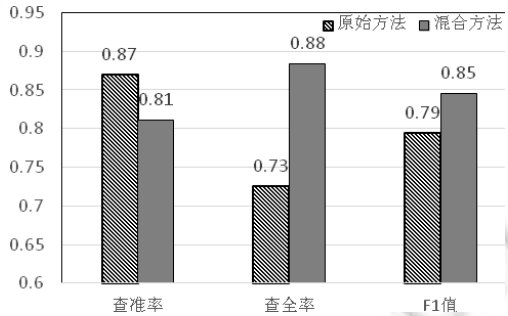

Fig.8 Compare results between orginal functionand hybird function for dividing samples

图 8 原始样本划分与混合样本划分方法分类效果对比

综合图 7 和图 8,一方面我们分析查全率上升和查准率下降的原因,由于类别阈值降低,divide-by-2 样本划分方法在模型中使用更加频繁,使得每个分类器在训练时的正负样本规模差距减小,降低了分类模型中分类器的数据倾斜程度,从而提高了查全率.查准率的下降是由传递误差累计导致.由于层次分类模型仅当上层父节点输出为真时,制品样本才会在下一层子节点进行预测,被上一层分类器拒绝的样本无法传递给子节点分类器参与预测,即,父类别的误差累计到了子类别中.随着 divide-by-2 对类别分组更加频繁,分类器的层次也在增加,引起的误差累计越大,导致整体查准率有所下降.

另一方面,*F*-measure 相比于查准率和查全率更具有综合评价的参考意义,是评价分类器的综合指标.本文采用的微平均评价方式,实例数量较多的类别在微平均过程中占有更大比重,而该部分类别一般处在较低深度层次,累计误差对其查准率影响较小,从而整体查准率下降程度小于查全率的上升.综合两者的 *F*-measure 得到了提高,这也表明该方法的有效性.因此在使用混合样本划分策略时,需要对兄弟类别阈值的设定进行权衡,以达到最优的分类效果.

3.6 其他案例分析

通过对比制品人工分类和采用本文方法进行制品自动分类所花费的时间,本文初步分析和评价了方法的执行效率.对 1 000 个 Chef 脚本制品进行人工分类标注花费了近 20 个小时,而采用本文方法则仅仅花费了 5 分钟即可完成,同时也具有较高的准确率(0.87).

我们采用 mysql 作为关键词进行 Puppet 和 Ansible 制品库的搜索,获取了 498 个软件制品.但是除了其中 103 个确实是 mysql 相关的制品外,其余制品皆与之无关,例如 postfix,collectd,zabbix 等等,因此搜索精度仅有 0.21.采用本文方法,我们能够得到 *mysql⊂rmde⊂database* 类别下的 153 个 mysql 相关制品,其中 137 个经过人工确认为正确的,具有 0.89 的准确率.我们使用几个案例来证明本文方法能够帮助用户缩小搜索范围,并完成更为准确合理的分类.

1) 基于 mysql 的关键词搜索结果包含了 Puppet 的一个脚本制品 puppet/zabbix,它是用来支持监控系统 Zabbix 的安装与管理操作,与 mysql 无关.我们的分类方法则将该制品划分到 *zabbix⊂monitoring⊂system* 类别下,显然更加合理;
2) Ansible 脚本库中的一个 mysql 相关制品 cranework.mysql 被错误的标记为 Web,本文分类方法则能够正确的将其划分到 *mysql⊂sql⊂database* 类别下;
3) Ansbile 脚本库中的制品 brisho.sSMTP 缺少对应的标签信息,这个制品能够为邮件服务器提供管理功能.本文方法通过自动分类将其划分到 *email⊂server* 类别,符合该制品的功能和特征.

上述案例分析证明,本文方法能够在制品分类的准确性和合理性方面提供有效的帮助.

3.7 讨 论

实验和分析结果表明:本文方法对于 CMT 脚本制品的结构化层次分类具有较好的表现,能够帮助用户更加准确地查找期望的目标制品.但是存在以下方面影响本文方法的有效性.

从外部来看,影响本文方法有效的因素主要在于方法是否具有普适性.首先,影响方法普适性的因素来自于与本文方法所选取的样本和数据是否具有代表性.由于脚本制品是 CMT 工具相关的,因此很难判定本文方法对于所有 CMT 工具的制品都适用.本文通过尽量选取最为主流的和具有代表性的 CMT 工具制品来降低这一因素的影响,主要采用 Puppet,Ansible,Chef 的超过 14 000 例 CMT 制品作为数据集.同时,本文将在后续工作中通过扩大数据规模和 CMT 工具类型范围来进一步验证工作的通用性.其次,制品标签是用户相关的,具有随机性,这对于构造层次化分类体系产生影响.本文从两个方面来应对这一问题:一方面,我们过滤掉不常用的标签来提高标签的正确性;另一方面,通过引入 Stack Overflow 的同义词库来进行同义标签的标准化.Stack Overflow 作为全世界最为活跃和最受欢迎的计算机领域问答社区积累了上千条同义词记录,具有实际的借鉴意义.

在内部影响因素方面,本文方法目前只采用了制品的名称和描述作为构建分类器的文本特征信息,并没有考虑诸如脚本作者和使用说明等其他更多类型的文本信息.除此之外,本文方法在构建层次化分类时只是基于标签的共现性(co-ocurrence)来分析他们之间可能存在的层次包含关系,并没有分析标签的语义信息.尽管实验结果验证了方法的有效性,但是本文后续工作将考虑引入更多的特征信息以及标签的语义信息.

4 相关工作

本文是首个针对配置管理工具制品进行层次化分类的工作,其相关工作主要包括软件分类和配置管理两个方面.

4.1 配置管理

针对如何提高 CMT 制品质量的问题,相关工作提出了解决方法,包括静态验证[31]和自动化测试[32,33]等.Collard 等人[31]提出了一种基于静态验证的方法来检测 Puppet module 的执行结果是否具有确定性(determinism).Hummer 等人[32]提出了一种基于黑盒的自动化测试方法来检测 Chef cookbooks 的执行操作是否具有幂等性(idempotence).该方法在执行 Chef cookbooks 的过程中,通过分析目标系统的变化来分析判断当前的 CMT 制品(cookbooks)是否具有幂等性.创建可靠的自动化系统配置脚本十分困难,Hanappi 等人[33]针对这一问题提出了基于模型的自动化测试框架,用来检测系统配置是否能够最终收敛到一个稳定的、期望的目标状态.Sharma[34]对 Puppet 脚本制品源代码进行分析,挖掘脚本编写设计和实现过程中容易出现的代码错误和不规范,以案例分析形式总结出 20 条最佳范例,指导脚本制品的开发.

在 DevOps 知识获取与制品管理方面也存在着相关的研究工作.Leymann 等人[35]提出了基于众包(crowdsourcing)与自动爬取相结合的方法来获取、管理和使用 DevOps 知识.通过集成知识库、基于谓词逻辑的查询方法和策略框架(policy framework),该方法提出了一整套用以组织、存储、查询和使用 DevOps 领域知识的途径.特别的,在 DevOps 工具、制品和服务分类方面提出了一种系统化的方法,该方法基于人工提出的分类体系进行 DevOps 相关实体(entity)的类别划分.

与本文工作不同,以上多数工作主要关注 CMT 制品的质量问题.虽然 Leymann 等人的工作涉及到 CMT 制品的分类管理,但是他们采取的是一种基于人工建立分类体系的方法,不同于本文提出的基于标签自动建立层次化分类体系.

4.2 软件制品分类

软件制品分类的相关工作可以分为两类,即,基于内部特征(internal feature)的分类方法和基于外部特征(external feature)的分类方法.

基于内部特征的分类方法主要基于软件源代码、注释和 API 调用信息实现分类.Ugurel[15]提出的方法分析程序源码结构并抽取标识符,然后结合注释关键字组成文档代表该软件,最后利用支持向量机(support vector machine,简称 SVM)方法将软件划分到对应主题和语言类别中.Linares[25]发现,对于部分软件(如基于 Java 的软件)可以收集并分析软件制品对第三方平台 API 的调用信息.基于此,他提出通过 API 反映的软件功能预测其类别的方法.CMT 制品领域语言相关,不适于采用基于内部特征和代码分析的方法进行分类.

基于外部特征的方法通过挖掘软件制品的外部特征,如软件制品在资源库中的名称、描述等在线属性实现软件分类.Wang[36]对 Freecode 软件仓库中软件标签分析,基于共现性度量标签相似度,提出一种基于 k-means 算法的软件标签层次构建分类方法.Dumitru[37]从大量软件描述中提取相关特征,利用增量扩散聚类算法实现基于初始输入特征的交互式软件关联推荐.Wang[26]基于 SourceForge、Freecode 等资源库,将相同软件的描述和标签属性聚合,通过 SVM 等文本分类算法,将软件划分到预定义的层次类别体系中.与本文工作不同,该工作基于 SourceForge 预先定义的层次分类体系实现软件分类,而本文则通过分析挖掘 CMT 制品标签间的层次包含关系来自动构建分类体系,解决了 CMT 制品不存在预定义分类体系的问题.

总体而言,当前面向 CMT 制品的分类仍然需要人工进行,缺乏高效的分类和检索体系.同时,CMT 制品的源代码、API 信息等受领域特定语言限制难以抽取有效信息.因此,本文受软件制品分类启发,整合多个 CMT 制品资源库,分析 CMT 制品在线非结构化描述文档,实现 CMT 制品的层次分类.

5 总 结

互联网为软件开发与维护提供海量资源,有效提取、组织与管理资源对 DevOps 实践有重要意义.本文提出了一种基于描述文档对 CMT 制品进行层次分类的方法,能够实现对配置管理工具(CMT)的脚本制品进行自动化分类.该方法不依赖于 CMT 制品的脚本代码和领域特定语言,具有良好的分类效果和扩展性.本文方法首先基于制品标签提出了一种层次分类体系的自动构建方法,基于该方法,本文对超过 11 000 个制品建立了包含 90 个细粒度类别的多层次分类树.然后基于监督学习方法,本文建立并训练了一组分类器实现对脚本制品的自动分类.特别的,针对分类器训练过程中,正反样本划分划分存在的数据倾斜问题,本文还提出一种改进的混合样本划分模型,有效提升了整体层次分类效果.

下一步工作包括:一方面,从 CMT 制品的源代码、配置文件等角度提取制品技术特征,与描述文档互为补充,探索新的脚本制品分类模型;另一方面,获取 Saltstack、Cfengine[11]等更多配置管理工具的脚本制品,构建跨 CMT 的脚本制品知识库,研究充分利用 CMT 制品资源辅助软件开发的途径.

References:

[1] Hüttermann M. Infrastructure as Code. In: Proc. of the DevOps for Developers. Apress, 2012. 135–156. [doi: 10.1007/978-1-4302-4570-4_9]

[2] RightScale. 2016 State of the cloud report. 2016. http://www.rightscale.com/lp/2016-state-of-the-cloud-report

[3] Chef Supermarket. Repositories of chef. 2016. https://supermarket.chef.io/cookbooks/

[4] Puppet Forge. Repositories of puppet module. 2016. https://forge.puppetlabs.com

[5] Ansible Galaxy. Repositories of ansible role. 2016. https://galaxy.ansible.com/list

[6] OpenHub. Discover, track and compare open source. 2016. https://www.openhub.net/explore/projects

[7] SourceForge. Find, create, and publish open source software for free. 2016. https://sourceforge.net/

[8] Zabbix SIA. Zabbix, the enterprise-class monitoring solution for everyone. 2016. http://www.zabbix.com/

[9] Galstad E. Nagios, the industry standard in IT infrastructure monitoring. 2016. http://www.nagios.org/

[10] Fu W, Cheney J, Anderson P. An operational semantics for a fragment of the puppet configuration language. ArXiv preprint arXiv: 1608.04999, 2016.

[11] CFEngine, Automate large-scale, complex and mission critical IT infrastructure. 2016. https://cfengine.com/

[12] Goldsack P, Guijarro J, Loughran S, Coles A, Farrell A, Lain A, Murray P, Toft P. The SmartFrog configuration management framework. ACM SIGOPS Operating Systems Review, 2009,43(1):16−25. [doi: 10.1145/1496909.1496915]

[13] Kawaguchi S, Garg PK, Matsushita M, Inoue K. Mudablue: An automatic categorization system for open source repositories. Journal of Systems and Software, 2006,79(7):939−953. [doi: 10.1016/j.jss.2005.06.044]

[14] Tian K, Revelle M, Poshyvanyk D. Using latent dirichlet allocation for automatic categorization of software. In: Proc. of the 6th IEEE Int'l Working Conf. on Mining Software Repositories. IEEE, 2009. 163−166. [doi: 10.1109/msr.2009.5069496]

[15] Ugurel S, Krovetz R, Giles CL. What's the code? Automatic classification of source code archives. In: Proc. of the 8th ACM SIGKDD Int'l Conf. on Knowledge Discovery aAnd Data Mining. ACM Press, 2002. 632−638. [doi: 10.1145/775047.775141]

[16] Manning C, Raghavan P. Introduction to Information Retrieval. Online Edition, Cambridge University Press, 2009. 118−120. [doi: 10.1017/CBO9780511809071]

[17] Wang T, Wang H, Yin G, Ling CX, Li X, Zou P. Mining software profile across multiple repositories for hierarchical categorization. In: Proc. of the 29th Int'l Conf. on Software Maintenance. 2013. 240−249. [doi: 10.1587/transinf.2014EDP7007]

[18] Cookbooks. About cookbooks. 2016. https://docs.chef.io/cookbooks.html

[19] Vural V, Dy JG. A hierarchical method for multi-class support vector machines. In: Proc. of the 21st Int'l Conf. on Machine Learning. ACM Press, 2004. 105. [doi: 10.1145/1015330.1015427]

[20] StackOverflow tag synonyms. http://stackoverflow.com/tags/synonyms/

[21] Liu K, Fang B, Zhang W. Ontology emergence from folksonomies. In: Proc. of the 19th ACM Int'l Conf. on Information and Knowledge Management. ACM Press, 2010. 1109−1118. [doi: 10.1145/1871437.1871578]

[22] Silla Jr CN, Freitas AA. A survey of hierarchical classification across different application domains. Data Mining and Knowledge Discovery, 2011,22(1-2):31−72. [doi: 10.1007/s10618-010-0175-9]

[23] Xue GR, Xing D, Yang Q, Yu Y. Deep classification in large-scale text hierarchies. In: Proc. of the 31st Annual Int'l ACM SIGIR Conf. on Research and Development in Information Retrieval. ACM Press, 2008. 619−626. [doi: 10.1145/1390334.1390440]

[24] He L, Jia Y, Han WH, Tan S, Chen ZK. Research and development of large scale hierarchical classification problem. Chinese Journal of Computers, 2012,35(10):2101−2115 (in Chinese with English abstract). [doi: 10.3724/sp.j.1016.2012.02101]

[25] Linares-Vásquez M, McMillan C, Poshyvanyk D, Grechanik M. On using machine learning to automatically classify software applications into domain categories. Empirical Software Engineering, 2014,19(3):582−618. [doi: 10.1007/s10664-012-9230-z]

[26] Wang T, Wang H, Yin G, Yang C, Li X, Zou P. Hierarchical categorization of open source software by online profiles. IEICE Trans. on Information and Systems, 2014,97(9):2386−2397. [doi: 10.1587/transinf.2014edp7007]

[27] Python3. Extensible library for opening URLs. 2016. https://docs.python.org/2/library/urllib2.html

[28] Leonard Richardson. Beautiful soup. 2016. https://www.crummy.com/software/BeautifulSoup/

[29] Sun A, Lim EP. Hierarchical text classification and evaluation. In: Proc. of the IEEE Int'l Conf. on Data Mining (ICDM 2001). IEEE, 2001. 521−528. [doi: 10.1109/icdm.2001.989560]

[30] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. Scikit-Learn: Machine learning in python. Journal of Machine Learning Research, 2011,12:2825−2830.

[31] Shambaugh R, Weiss A, Guha A. Rehearsal: A configuration verification tool for puppet. In: Proc. of the 37th ACM SIGPLAN Conf. on Programming Language Design and Implementation. ACM Press, 2016. 416−430. [doi: 10.1145/2980983.2908083]

[32] Hummer W, Rosenberg F, Oliveira F, Eilam T. Testing idempotence for infrastructure as code. In: Proc. of the ACM/IFIP/USENIX Int'l Conf. on Distributed Systems Platforms and Open Distributed Processing. Berlin, Heidelberg: Springer-Verlag, 2013. 368−388. [doi: 10.1007/978-3-642-45065-5_19]

[33] Hanappi O, Hummer W, Dustdar S. Asserting reliable convergence for configuration management scripts. In: Proc. of the 2016 ACM SIGPLAN Int'l Conf. on Object-Oriented Programming, Systems, Languages, and Applications. ACM Press, 2016. 328−343. [doi: 10.1145/2983990.2984000]

[34] Sharma T, Fragkoulis M, Spinellis D. Does your configuration code smell? In: Proc. of the 13th Int'l Workshop on Mining Software Repositories. ACM Press, 2016. 189−200. [doi: 10.1145/2901739.2901761]

[35] Wettinger J, Andrikopoulos V, Leymann F. Automated capturing and systematic usage of devops knowledge for cloud applications. In: Proc. of the 2015 IEEE Int'l Conf. on Cloud Engineering (IC2E). IEEE, 2015. 60−65. [doi: 10.1109/IC2E.2015.23]

[36] Wang S, Lo D, Jiang L. Inferring semantically related software terms and their taxonomy by leveraging collaborative tagging. In: Proc. of the 28th IEEE Int'l Conf. on Software Maintenance (ICSM). IEEE, 2012. 604−607. [doi: 10.1109/ICSM.2012. 6405332]

[37] Dumitru H, Gibiec M, Hariri N, Cleland-Huang J, Mobasher B, Castro-Herrera C, Mirakhorli M. On-Demand feature recommendations derived from mining public product descriptions. In: Proc. of the 33rd Int'l Conf. on Software Engineering (ICSE). IEEE, 2011. 181−190. [doi: 10.1145/1985793.1985819]

附中文参考文献:

[24] 何力,贾焰,韩伟红,谭霜,陈志坤.大规模层次分类问题研究及其进展.计算机学报,2012,35(10):2101−2115. [doi: 10.3724/sp.j. 1016.2012.02101]

徐培兴(1991－),男,山东聊城人,硕士生,主要研究领域为分布式计算,云计算.

陈伟(1980－),男,博士,副研究员,CCF 专业会员,主要研究领域为软件工程,分布式计算,服务计算,云计算.

吴国全(1979－),男,博士,副研究员,CCF 专业会员,主要研究领域为网络分布计算,软件工程.

高楚舒(1978－),男,博士,助理研究员,主要研究领域为软件工程,服务计算.

魏峻(1970－),男,博士,研究员,博士生导师, CCF 高级会员,主要研究领域为分布式计算,软件工程.