基于在线属性聚合的海量软件层次分类

王 涛[1] 王怀民[1] 尹 刚[1] 李 翔[1] 杨 程[1] 邹 鹏[2]

[1](国防科学技术大学计算机学院 并行与分布处理国家重点实验室 长沙 410073)

[2](装备学院 北京 101400)

摘 要 互联网规模的软件资源库正从根本上改变传统的软件开发模式,资源库中海量软件的高效层次分类对基于互联网资源的软件开发具有重要意义.传统软件分类方法基于软件源代码或字节码实现粗粒度的扁平分类,并且只在小规模数据集上进行了验证.文中提出了一种基于软件在线属性聚合的层次分类方法,设计了一个层次分类框架,基于跨资源库软件在线描述和标签的加权聚合,实现对海量软件的高效层次化分类.文中在超过 18000 个开源软件上进行交叉验证,实验结果表明文中提出的在线属性加权聚合方法能显著提高软件分类效果.在粗粒度扁平分类下文中方法能够达到基于源代码/字节码分类近似的性能,而且,与相关工作比较,文中方法实现了涵盖 123 个更细粒度类别的层次化分类,能够更有效地对海量软件进行分类.

关键词 软件资源库;开源软件;层次分类;在线属性

中图法分类号 TP311 **DOI 号** 10.3724/SP.J.1016.2013.02007

Hierarchical Software Categorization Based on Aggregation of Online Attributes

WANG Tao[1] WANG Huai-Min[1] YIN Gang[1] LI Xiang[1] YANG Cheng[1] ZOU Peng[2]

[1](*National Key Laboratory of Parallel and Distributed Computing, School of Computer Science, National University of Defense Technology, Changsha* 410073)

[2](*Academy of Equipment, Beijing* 101400)

Abstract The Internet-scale software repositories are fundamentally changing the traditional paradigms of software development. Efficient categorization of the massive software projects in these repositories is of vital importance for Internet-based software development. Traditional classification approaches do coarse-grained and flat categorization by analyzing source code or byte code, and most of them are only verified on relatively small collections of software projects. In this paper, we propose an efficient hierarchical categorization approach based on the aggregation of the software online attributes and design a hierarchical categorization framework. Based on the weighted aggregation of software descriptions and tags across multiple repositories, we categorize the massive software hierarchically. Extensive experiments are carried out on more than 18,000 software projects. The results show that significant improvement can be achieved by using weighted aggregation of different online attributes. Compared to the previous work, our approach achieves/gains competitive performance with 123 hierarchical and finer-grained categories for which classification is much harder. In contrast to those using source code or byte code, our approach is more effective for large-scale categorization.

Keywords software repository; open source software; hierarchical categorization; online attribute

收稿日期:2013-06-26;最终修改稿收到日期:2013-09-01.本课题得到国家"八六三"高技术研究发展规划项目基金(2012AA011201)、国家自然科学基金(60903043)资助.**王 涛**,男,1984 年生,博士研究生,中国计算机学会(CCF)会员,主要研究方向为实证软件工程、开源软件挖掘.E-mail:taowang.2005@outlook.com.**王怀民**,男,1962 年生,教授,博士生导师,中国计算机学会(CCF)高级会员,主要研究领域为软件工程、虚拟计算环境与中间件.**尹 刚**,男,1975 年生,博士,副研究员,中国计算机学会(CCF)会员,主要研究方向为数据挖掘与软件工程.**李 翔**,男,1988 年生,博士研究生,主要研究方向为软件工程与数据挖掘.**杨 程**,男,1991 年生,硕士研究生,主要研究方向为软件工程与数据挖掘.**邹 鹏**,男,1957 年生,教授,博士生导师,主要研究领域为网络与信息安全.

1 引 言

互联网规模的软件资源库如 SourceForge、Ohloh 以及 RubyForge 等都聚集了海量的开源软件资源,这些开源软件资源正从根本上改变传统的软件开发活动.传统软件开发通常基于组织内部的本地资源库,开发活动通常局限在开发团队或者组织内部.随着互联网软件资源库不断增长,可自由获取的高质量软件资源不断积累,开发人员可利用的不再仅仅是组织内部的本地资源库,而是整个互联网上的软件资源库,软件开发活动正演变成一个全球协作的过程.借助互联网软件资源库,开发人员可以查找问题解决方案、学习他人最佳实践等.比如,通过资源库搜索查找高质量可复用组件[1]、发现特定领域的新技术趋势[2-3]、从相关系统学习解决方案[4]、分析相似软件来预测和修改软件缺陷[5],或者找到相关领域专家进行交流讨论[6].那么,从互联网规模的软件资源库中高效准确地检索到需要的软件非常重要.

层次分类通过类别划分大大缩小搜索范围,从而被认为是一种大规模数据快速准确检索的有效方法[7].这种层次分类机制在 SourceForge、RubyForge 等大型资源库中被广泛使用,通过与通用关键字搜索相结合实现高效的资源检索.但是,这些资源库中的软件利用人工分类,而且有相当大比例的软件没有进行分类[8].此外,在 Ohloh、Freecode 等开源资源库则完全没有分类系统.

在软件自动分类领域已有很多的相关研究,主要通过分析软件程序实现分类.其中,文献[9-11]通过源代码解析抽取相应的属性,用这些属性组成的文本来代表相应软件,然后利用文本分类方法进行分类.McMillan 等人[12-13]提出一种基于程序 API 调用信息的软件分类方法,通过分析第三方 API 对应的类别信息来预测程序所属类别.因为 API 调用信息既能从源代码中获得,也能从字节码中得到,因此这种方法对那些没有源代码的软件也能进行自动分类.目前这些工作主要研究扁平分类且类别粒度都比较大(仅定义了 20 个左右的类别),如“Internet”等.在海量软件中,如此大粒度的类别对缩小软件搜索范围还远远不够.如在 SourceForge 中,“Internet”类别下的软件数目仍然超过 35 350 个.此外,在这些研究中,绝大部分工作仅仅在很少数量的软件上进行了实验验证.但是,考虑到在互联网资源库中软件的数量和软件本身的复杂性(在 Ohloh 资源库中有超过 400 000 个软件,所使用的编程语言超过 100 种),如何对这些资源库中的海量软件进行高效自动的细粒度分类是一个非常有挑战性的问题.

本文提出了一种层次分类方法,充分利用软件在线属性实现海量软件的高效自动分类.互联网资源库中的软件通常利用软件简介或者标签进行概括描述,这些描述和标签等在线属性比较全面地反映了软件的功能和技术特征,因此为我们进行软件分类提供了新的源数据.以数据库管理系统 MySQL 为例,其在 SourceForge、Ohloh 和 Freecode 中的描述和标签/类别属性见表 1 和表 2.

表 1 MySQL 在资源库中的描述

资源库	软件描述
SourceForge	MySQL is a well-known relational database manager used in…. Offering a wide range of features and highly scalable performance. MySQL is a good choice for any situation requiring a database.
Ohloh	MySQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation.
Freecode	MySQL is a widely used and fast SQL database server. It is a client/server implementation that consists of a server daemon (mysqld) and many different client programs/libraries.

表 2 MySQL 在资源库中的标签/类别

资源库	软件标签/类别
SourceForge	Database Engines/Servers
Ohloh	db, acid, database server, rdbms, odbc, database, dbi, jdbc, server, sql, software development, replication, mysql, dbms, …
Freecode	Database, Database Engines/Servers

源代码标识符和 API 调用信息通常反映软件在实现层类或包的细粒度特征.与此不同,软件描述和标签强调整个软件更高层的技术或功能特征.如表 1 和表 2 所示,MySQL 的软件描述和标签/类别等在线属性概括了软件的重要特征.同时,软件描述和标签以自然语言形式给出,绝大部分互联网软件资源库中的软件都有此类数据.因此,这类数据与编程语言无关且更容易获得,从而能够基于这类数据实现更为高效和通用的软件分类.

我们首先基于 SourceForge 定义的分类层次构建了一个包含超过 120 个类别的 4 层分类体系.然后,设计了一种基于支持向量机(SVM)等算法的分类框架,实现了基于软件在线数据的层次分类.通过广泛的实验分析对比了不同的分类方法并深入讨论

了利用跨资源库在线属性聚合的方法来优化软件层次分类.具体而言,本文的加权聚合方法,通过将多个资源库中不同的软件在线属性(本文主要针对描述和标签)进行聚合,并对不同数据根据其重要性赋予不同权重,大大提高了软件分类准确度.

本文第 2 节介绍软件分类和软件在线属性挖掘的相关工作;第 3 节阐述软件层次分类方法;第 4、5 节给出相关实验并对实验结果进行分析;第 6 节对全文工作进行总结与展望.

2　相关研究

目前,对软件分类、软件资源库挖掘方面的研究较多,本节我们重点介绍在软件自动分类和软件在线属性数据挖掘方面的相关工作.

2.1　软件自动分类研究

在软件自动分类方面,根据分类所依赖的源数据,当前的研究工作主要可以分为两类.

第 1 类是基于源代码标识符和注释的软件分类方法,主要研究包括文献[9-11].在这些工作中,每个软件被看作是一个由源代码标识符和注释组成的文档,然后利用文本分类方法实现对这些软件的分类.这些研究的主要区别在于所采用的类别定义以及机器学习方法.MUDABlue[9] 和 LACT[10] 首先从源代码中自动构建分类类别,然后分别采用 LSA 和 LDA 的方法来对软件进行分类;文献[11]则抽取源代码标识符和注释作为软件对应文档,然后利用 SVM 方法将软件分类到预定义的主题和语言类别中.这一类工作只能对有源代码的软件进行分类.

第 2 类工作主要基于其它软件信息如 API 调用信息来实现软件分类[12-13].由于很多软件无法获得源代码,Linares-Vsquez 等人[13] 提出了一种新的基于 API 调用信息的软件分类方法.其基本思想是:外部 API 和方法通常是基于功能组合在一起的,因此这些 API 和方法能够反映调用者的功能,从而能够用来预测调用者的类别.API 调用信息既能通过源代码分析得到,也能通过字节码分析获得,因此这种方法对无法获得源代码的软件也能够进行分类.

目前,绝大部分软件分类研究需要对软件源代码或者字节码进行分析.然而,在互联网软件资源库中通常积累了海量的软件资源,同时这些软件代码非常复杂,要基于源代码或字节码分析的方法对软件资源库中的海量软件进行分类面临很大的挑战.同时,目前这些方法都是扁平分类且类别粒度较大.不同于此,本文探索了基于软件在线属性的层次分类方法.

2.2　软件在线属性数据挖掘

随着越来越多的软件项目被发布到互联网资源库中,软件在线数据也不断积累,很多研究人员开始关注此类数据,对其展开研究.

Dumitru 等人[2] 及 Yu 等人[14] 对 Softpedia 中的软件在线特征描述数据进行研究以辅助领域分析.Dumitru 等人提出了一种增量扩散聚类算法,实现从大量软件特征描述中发现领域相关特征,并根据领域分析人员的初始输入进行分析并做关联特征推荐.McMillan 等人[15] 更进一步,通过对源代码中方法调用链的分析,将文献[2]中挖掘的特征定位到具体的实现模块.

此外,软件协同标签被互联网资源库广泛使用来描述软件特征、组织海量软件资源.Li 等人[16] 和 Wang 等人[17] 对资源库中的软件标签进行了深入研究.文献[17]作者对 Freecode 中的软件标签进行分析,基于同现性对标签相似度进行度量,提出一种 k-中心聚类算法来构建开源软件的标签层次.此外,他们还对 SourceForge 中的软件类别、编程语言、许可证等软件标签进行分析以查找相似软件[18].

这些工作对软件在线属性进行了分析和研究.但是目前的研究通常聚焦于某一个特定的资源库,没有将不同的在线属性关联起来.本文将多个资源库中的软件描述和协同标签等不同的属性数据聚合起来,实现软件的层次分类.

3　软件层次分类

软件资源库中积累了海量的软件,这些软件通常附带了相应的软件描述和软件标签等在线属性数据,为进行软件分类提供了源数据.本文基于这些数据实现自动层次化分类方法.本节我们介绍基于软件在线属性聚合的层次分类处理流程,然后对软件在线属性、层次分类器训练与预测进行介绍.

3.1　软件层次分类流程

基于在线属性聚合的层次分类主要分为数据获取与预处理、分类层次构建、软件在线属性聚合以及分类器训练与部署 4 个部分,其处理流程如图 1 所示.

(1)数据获取与预处理.利用网络爬虫将软件在各个资源库中的主页爬取下来,然后通过 HTML 解析工具抽取出各软件的描述、类别和标签等在线

属性,并进行停用词剔除、词干提取等预处理操作.我们最终得到每个软件预处理后的在线属性,并将这些在线属性作为软件对应的文档.

(2)分类层次构建.互联网上有超过百万的开源软件项目,因此粗粒度的分类对于缩小检索范围还远远不够.本文基于在开源社区中广泛采用的分类体系,根据所涵盖的软件主题以及主题之间的关系将不同粒度的类别组织成一个层次分类结构.

(3)在线属性加权聚合.不同的资源库中同一软件的在线属性并不完全一样,这些属性从不同角度和层面反映软件的功能和技术等特征,通过聚合这些在线属性能够更全面地概括一个软件.本文提出了一种加权聚合方法,根据不同类型在线属性的特点以及其在区分软件类别的重要度,将软件在多个社区的不同属性进行加权聚合.

(4)分类器构建与部署.利用聚合属性本文基于支持向量机模型(SVM)等分类方法、采用"自顶向下"的方法构建层次分类系统.基于构建的分类器,我们可以将其部署到软件资源库中,实现基于软件在线属性的自动分类.

层次分类流程图如图1所示.

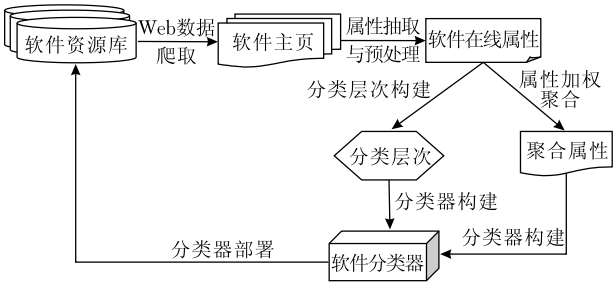


图1 基于软件在线属性聚合的层次分类流程图

3.2 软件在线属性

3.2.1 软件在线属性数据

软件在线属性包括多种类型的数据,本文重点研究软件在线描述、类别和标签这3类属性数据.在软件资源库中,每个软件在发布时通常会附带相应的软件描述,对软件的主要功能等进行概括性介绍,从而帮助使用者了解该软件.第1节表1列出了MySQL在开源资源库中的描述,从这些描述中我们可以看到"SQL"、"database management system"和"Oracle"等与数据库相关的词,基于这些词汇可以判定该软件的类别.

除软件描述外,软件标签被广泛应用于对海量软件资源的标注.在Ohloh和Freecode中,所有注册用户都可以根据他们对相应软件的认识和了解为资源库中的软件添加新标签、删除或者修改已有的标签.第1节表2列出了MySQL在Ohloh和Freecode中的标签,这些标签聚集了不同使用者对MySQL的认识和了解,从不同的角度反映了MySQL的特征,为进行软件分类提供了有效信息.

软件描述和软件标签都从较高层次反映软件的整体特征,与源代码标识符以及API调用信息是互补的.源代码标识符和API名称等在方法、类或者包的粒度上反映了软件的特征,而软件描述或者标签则以整个软件为粒度概括了软件的重要技术或者功能特征.基于这个思想,我们通过分析挖掘软件在线描述与标签来对软件进行自动分类.

3.2.2 在线属性加权聚合

尽管软件描述和软件标签都从整体上反映软件的技术或者功能特征,但这两类在线属性存在差异.软件描述通常是软件管理者或者提交者为软件能够吸引用户而写的.在软件描述中,除了反映软件功能或技术特征之外还有很多其它无关词,这些词会对软件分类造成干扰.软件标签的标注者既包括软件提交者或管理者,也包括软件使用者、软件维护人员等.他们为软件添加标签以反映软件的某些重要特征,这些特征涵盖了软件的功能和技术特征.因此,除一些拼写错误或者特殊的词,软件标签通常具有更好的质量.同时,这两类在线属性也具有一定的互补性,两者的聚合能够更全面地反映软件特征.但是,由于每个软件的标签通常较少而软件描述相对更长,将两者简单合并会导致标签的权重降低,进而削弱软件标签在分类中的作用.

为了平衡两者在软件分类中的影响,我们设计了一种加权策略,通过增加标签出现次数来提高软件标签的权重.其基本思路是:软件描述和软件标签都是从整体上对软件特征进行概括,因而可以认为一个软件的描述及其标签在区分一个软件的类别时具有等价的效果.基于此,我们设计了一种方法使得软件描述归一化词频之和与所有标签的归一化词频之和相近,从而保证两者的总体权重相近.具体做法是在将软件描述和标签进行合并之前,我们首先将软件的标签复制多份然后再合并,复制份数由软件描述与软件标签词的比例来确定.具体的复制次数由式(1)确定.

$$\delta=\alpha\times\sqrt{\frac{\sum_{k}t_{kj}}{\sum_{m}\hat{t}_{mj}}} \tag{1}$$

式(1)中,t_{kj}表示词t_k在软件j的两类在线属性中出

现的次数，$\hat{t}_{mj}$ 表示标签 $\hat{t}_m$ 在软件 j 中出现的次数. 在实际数据中，部分软件的标签数很少（仅1～2个），不足以全面涵盖软件各方面的特征，难以达到和软件描述等价的效果. 对于此类软件，直接根据描述与标签数目比例进行复制会使该标签出现次数远大于其它描述词，导致其对分类的影响过大，影响分类模型的准确性. 因此，我们对软件描述长度与软件标签数的比例进行开方从而降低标签复制的过大影响. 此外，我们在公式中增加参数 α 对复制次数进行控制. α 为0表示不复制而直接进行简单合并，此时标签可以被看成是软件描述中的一个普通词. 随着 α 的增加，对标签的复制次数不断增大，相应的标签权重会不断增大. 下面我们将具体分析这种复制对权重的影响.

本文使用 TF-IDF 来表征一个词在区分软件类别时的重要性，每个词的 TF-IDF 值可根据式(2)来计算. 在式(2)中，t_{ij} 表示词 t_i 在软件 j 中出现的次数，其中 t_i 可以是一个描述中的词或者一个标签. n_i 表示包含词 t_i 的软件个数，N 表示软件总数. 式(2)第1部分是词 t_i 在软件 j 中的归一化词频，第2部分是词 t_i 的逆向文档频率.

$$w_{ij}=\frac{t_{ij}}{\sum_k t_{kj}}\times\log\frac{N}{n_i} \tag{2}$$

从式(2)可以看出，在合并时对软件标签进行复制能够增加标签的归一化词频同时降低软件描述中词的归一化词频，但是这种复制不会改变标签或描述词的逆向文档频率. 因此，综合起来，利用标签复制的方法能够提高标签的整体权重同时降低描述词的权重.

3.3　层次分类方法

3.3.1　分类层次定义

本文将分类层次建模为一个树结构，父子节点之间是一种“包含”关系，这种关系具有反对称性、反自反性和传递性. 该分类结构有一个根节点“Root”将所有的第一层节点关联起来，除“Root”节点外所有类别都可以包含1个、多个子类或者0个子类，形成相应的子树.

图2给出了“Multimedia”对应的子树示例. 其中，“Multimedia”节点将所有子类别关联起来构成一棵树，其第一层节点如“Video”、“Sound/Audio”类分别构成下一级的子树. 在我们定义的类别层次中，对某些软件其最细粒度的类别可能不是叶节点而是内部节点. 同时，一个内部节点（如图2中的“Players”）可以只有一个子类别. 因为不强制要求所有软件都被划分到叶节点，同时所有软件都应尽可能被划分到更细粒度的类别中，所以某些类别只有一个子类是有意义的. 如图2，对于被划分到“Players”的所有软件，我们会继续测试其是否属于更细粒度的“MP3”. 如果属于的话，我们就将其划分至“MP3”，否则将其划分至“Players”.

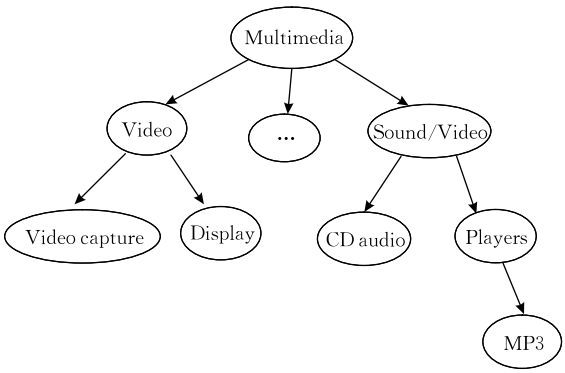


图2　“Multimedia”子树示例图

在该层次定义中，我们不允许两个类别节点间存在多条路径，即同一个类别不能同时属于两个不同的父类（多重从属关系）. 这主要是为了后续方法说明的简单清晰. 在实际的软件类别体系中这类情况是存在的，而且本文的方法也能够较好地解决这一问题，具体细节我们将在分类模型构建与预测章节进行分析.

3.3.2　分类模型构建与预测

层次分类模型的构建主要可以分为“Big-bang”和“Top-down”两大类方法[19]. “Big-bang”面向整个类别层次学习一个包含所有类别的分类器，这种方法能够有效降低分类器模型的规模. 但是，由于层次类别总数较大，因此很难构建一个统一的分类器实现对所有类别的准确分类. 不同于此，“Top-down”方法为每个类别构建一个单独的分类器，并在预测时针对每个类别单独进行测试. 文献[20]通过实验证明“Top-down”方法要优于“Big-bang”的方法. 因此，本文采用“Top-down”的层次分类模型构建方法.

在对每个节点构建分类器时主要需要解决两个问题：分类算法的选择以及正反训练样例的构建. 目前有很多分类器构建方法如支持向量机(SVM)、最近邻(kNN)以及贝叶斯(Bayes)方法等. 其中 SVM 方法在前人的工作中被广泛应用于软件分类并被证明是最有效的方法[12,21]，本文将对这几类方法进行分析对比. 正反样例的选择，对于某个给定的类别，其正例样本为该类别及其子类下的所有样本，反例

为其兄弟节点及对应所有子节点下的样本.对于没有兄弟节点的类别(如图2中的"MP3"),我们回溯到其有兄弟节点的最近祖先节点,将该祖先节点其它子节点下的样本作为其反例样本.以"MP3"为例,其反例为节点"Players"的兄弟节点"CD Audio"下的所有样本.这种正反样本的选择方法能够降低层次分类中严重的"数据倾斜"带来的问题[20].通过这种方法我们为每一个类别构建对应的正反样本并训练分类器.

本文的层次分类问题是一个"非强制叶节点预测"问题,即待测试软件最细粒度的类别不强制要求为叶节点.如图1,类别"CD Audio"和"Players"并不能完全涵盖"Sound/Audio",因此某个软件可能属于"Sound/Audio",但是不属于任何一个子类.对于一个给定的软件,和前面的训练过程类似,我们仍然采用"Top-down"的方法来进行预测.首先,在第一层所有分类上对该软件进行测试.如果其属于某个类别,则进一步测试该类别下的所有子类;否则,停止对该类别所有子类的测试.

这里的层次模型构建与预测方法针对的是3.3.1节所定义的分类层次.在实际的类别体系中如果存在多重从属关系,如类别C3同时为C1和C2的子类,那么在分类器构建过程中确定正反样本时,要分析一个属于C3类别的软件A是C1的正例样本还是C2的正例样本时,我们可以通过两种简答的方法进行解决.一种方法是根据软件A的其它类别来判断,如果软件A有一个标注类别是C1的父类,那么可以判断A应该为C1的正例样本;如果这种方法仍然无法判断,那么我们可以简单认定软件A同时为C1和C2的正例样本.在进行预测时,因为我们采用了"Top-down"的分类方法并为每一个类别构建一个分类器独立进行预测,因此在进行预测时多重从属关系和单重从属关系是一样的,无须特殊处理.

4 实验设计

4.1 实验数据集

本文以三大开源软件资源库SourceForge、Ohloh和Freecode为例对本文所提方法进行验证.

SourceForge是目前最大的开源社区之一,托管了超过300000个软件项目.SourceForge定义了一个包括4层共363个类别的分类层次,项目管理者在提交项目时可以选择相应的类别.Ohloh和Freecode是两个主要提供软件名录和分析服务的开源软件资源库,各收集了超过400000和45000个软件项目.Ohloh和Freecode采用了标签机制来对其中的软件进行组织和管理,注册用户能够对其中的软件标签进行添加和删除等操作.这3个资源库涵盖了两类典型的互联网资源库,本文基于这些资源库中的软件描述和标签进行实验.

本文通过以下3个步骤来构建实验数据集:

(1)软件主页爬取.利用网页爬虫从资源库爬取各软件项目主页,分析并抽取软件描述、软件标签以及软件类别等数据.

(2)跨资源库软件描述与标签聚合.同一个软件可能同时存在于不同资源库中,我们将这一类软件抽取出来,并将其来自不同社区的描述和标签数据合并起来作为我们的原始训练和测试集.

(3)软件数据预处理.对聚合后的原始项目数据,我们进行去除停用词、提取词干等预处理,然后去掉描述长度少于10的软件.同时,我们设定了一个阈值50来剔除拼写错误或者特殊的标签,只有标注超过50个软件的那些标签才保留下来作为有效标签.我们最终得到18032个软件和5429个不同的标签,该数据集的详细信息如表3所示,其中SF,Oh和Fc分别代表取自SourceForge、Ohloh和Freecode的数据.

表3 实验数据集

资源库	软件数目	平均描述长度	不同类别总数	平均类别/项目数	不同标签总数	平均标签/项目数
SF	18032	19.69	307	2.98	—	—
Oh	9813	20.84	—	—	5373	5.73
Fc	10357	25.17	—	—	940	4.85

实验数据集中每个软件都存在于SourceForge中且在SourceForge中被指定了类别.同时,这些软件至少在Ohloh或Freecode中的一个资源库中存在,且有至少一个有效标签.这些数据的软件描述长度、平均类别数和平均标签数等如表3.

4.2 分类层次构建

本文基于SourceForge预定义的分类层次来构建分类层次体系,类似分类层次也被广泛应用于其它开源社区如RubyForge等.基于SourceForge中的分类层次,本文按以下步骤构建层次分类体系:

(1)DAG消除.在SourceForge原始分类系统中,两个类别节点之间可以存在多条路径.为简化分类过程,我们只保留出现最频繁的一条路径.

(2)分类层次裁剪.我们将最相似的类别进行合并,然后将样本数量少于某阈值的类别删除并将对应类别下的软件提升到其父节点下.构建统一的

分类层次.在消除 DAG 并进行裁剪后,我们创建虚拟根节点将所有节点连接起来构建一棵统一的类别层次树.

在进行类别裁剪时,对不同层次的类别我们设定不同的阈值,从第 1 到第 4 层的阈值分别为 500,100,50 和 50.经过这 3 步处理,我们最终构建出包括 4 层共 123 个类的分类层次.构建的分类层次详细信息如表 4,其中,第 1 层共有"Multimedia"等 12 个类,而第 2、3 层分别包含 61 和 41 个类别,第 4 层有 9 个类.在 18032 个软件中,第 1 层类别下的平均正例样本数目为 2215.33.随着类别层次增加,相应类别下的正例样本数目迅速减少.

表 4　分类层次详细数据

分类层次	类别个数	平均正例样本数
1	12	2215.33
2	61	382.15
3	41	196.85
4	9	120.00

4.3　评价标准

目前大多数扁平分类工作通常采用正确率(precision)、召回率(recall)以及 F_1 值(F-measure)来评价分类的性能[19].本文采用修改的层次分类评价标准来评估层次分类性能,主要包括针对每个类别的层次正确率(hierarchical Precision,hP)、召回率(hierarchical recall,hR)和 F_1 值(hierarchical F-measure,hF),其具体定义如下:

$$hP=\frac{\hat{P}_i\cap\hat{T}_i}{\hat{P}_i},\ hR=\frac{\hat{P}_i\cap\hat{T}_i}{\hat{T}_i},\ hF=\frac{2\times hP\times hR}{hP+hR} \tag{3}$$

式(3)中,对每个类 i,$\hat{P}_i$ 表示预测类别为 i 以及 i 的所有子类的样本集合,$\hat{T}_i$ 是所有原始标注类别为类 i 及其所有子类的样本集合.本文采用的测度 hP、hR 和 hF 分别表示对每个类别在所有测试样本上的平均正确率、召回率和 F_1 值.为度量在所有类别上的平均正确率、召回率和 F_1 值,类似于层次分类相关工作[19]中所采用的测度,我们采用微平均的方法来度量.其具体定义如下:

$$Micro\text{-}hP=\frac{\sum_i(\hat{P}_i\cap\hat{T}_i)}{\sum_i\hat{P}_i},$$

$$Micro\text{-}hR=\frac{\sum_i(\hat{P}_i\cap\hat{T}_i)}{\sum_i\hat{T}_i},$$

$$Micro\text{-}hF=\frac{2\times Micro\text{-}hP\times Micro\text{-}hR}{Micro\text{-}hP+Micro\text{-}hR} \tag{4}$$

5　实验分析

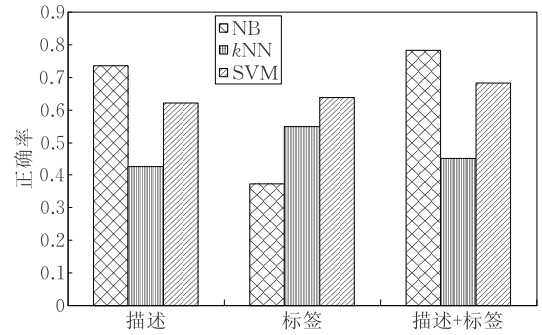
为验证本文提出的基于软件在线属性的层次分类方法以及加权聚合方法的有效性,我们设计了不同的实验.5.1 节我们将分析对比不同分类算法基于不同类型在线属性及其直接合并(不进行加权聚合)的分类效果;5.2 节我们将不同类型在线属性进行加权聚合并对比分析不同加权强度(通过式(1)中的参数 α 来控制)对分类结果的影响;5.3 节我们对比分析基于在线属性加权聚合的分类与基于 API 的分类方法.在 5.1 和 5.2 节实验中,我们采用 5 倍交叉验证的方法,实验数据被随机分为 5 等份,共进行 5 组实验,每组实验拿其中 4 份进行训练在剩下的 1 份上进行测试.这种方法能够保证实验数据集中的所有软件都能够作为测试项被测试一次,能够更好地验证方法的有效性.

5.1　基于软件描述与软件标签的分类

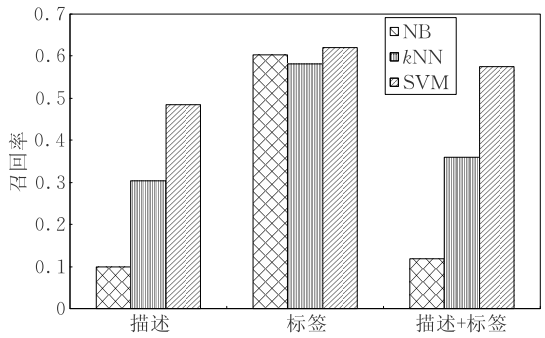
本节我们分别基于软件描述、标签以及描述和标签直接合并的数据设计实验来测试本文提出的层次分类方法.第 1 组实验我们首先将来自 SourceForge、Ohloh 和 Freecode 的同名软件的描述进行合并,然后构建分类器并进行测试;第 2 组实验我们将软件在 Ohloh 和 Freecode 中的标签作为软件文本进行测试;第 3 组实验,我们将同一软件在 3 个社区中的软件描述以及相应的标签进行直接合并作为软件文本进行训练和测试.

本节我们分别采用 Naïve Bayes、kNN 以及 SVM 方法作为基本分类器构建层次分类系统进行实验.前两种方法基于 Scikit-learn[22] 实现,SVM 基于 Lib-linear[23] 实现.具体实验结果如图 3 所示.

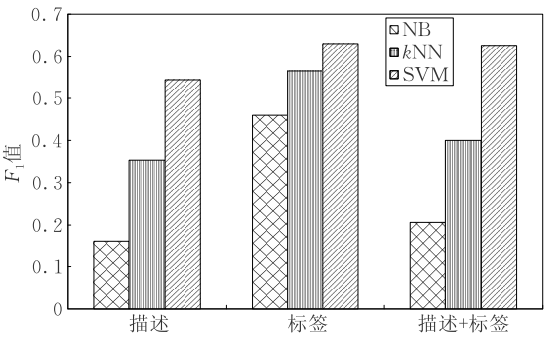
图 3(a)、(b)、(c)分别给出了不同分类方法基于不同在线属性分类的正确率、召回率及 F_1 值.对比 3 种不同的分类方法,从图 3 可以看出 Naïve Bayes 方法在基于描述以及描述与标签合并数据的分类正确率最高,但是召回率仅为 10%左右,但在基于标签数据时的分类正确率最差.基于 SVM 的层次分类在 3 类不同属性数据上能够获得优于 kNN、与 Naïve Bayes 相近的正确率,但是召回率要远优于 Naïve Bayes 和 kNN.综合考虑正确率和召回率,我们从图 3(c)的 F_1 调和均值可以看出,基于 3 类不同属性数据,SVM 都获得最优的效果.3 种方法的实验效果与当前相关工作的研究结果是一致的,这也是我们在后面的实验中选择 SVM 来构建



(a) 正确率



(b) 召回率



(c) F_1值

图 3 基于软件不同在线属性,利用 Naïve Bayes (NB)、kNN 和 SVM 方法的分类实验结果

层次分类系统的原因.

对比不同的软件在线属性对分类结果的影响,综合考虑分类正确率和召回率,从图 3(c)的 F_1 看,对 Naïve Bayes 方法和 kNN 方法而言,基于标签的分类结果要远好于基于描述或者描述与标签合并;对 SVM 方法,基于标签的实验结果要优于基于描述的实验,但与标签与描述合并结果相近.更具体的,以 SVM 方法为例,表 5 列出了 SVM 在基于 3 组数据实验的详细结果.

表 5 基于 SVM 方法在不同在线属性上的分类实验

软件简介	*Micro-hP*	*Micro-hR*	*Micro-hF*
描述	0.6213	0.4839	0.5440
标签	0.6375	0.6207	0.6290
描述+标签	0.6831	0.5746	0.6241

从表 5 中可以看出,基于软件描述时整体的正确率、召回率分别为 62.13%和 48.39%.而基于软件标签的实验,尽管相对于软件描述而言每个软件的标签要少很多,却能够获得更好的实验效果.从表中第 2 行可以看出,基于标签的实验中正确率和基于软件描述的结果差不多,但是其召回率却提高了约 13.68%,F_1 也提升了约 8.50%.分类效果得到提高的原因在于软件标签相对于软件描述具有更好的质量.以 MySQL 为例,其软件描述中包含很多词如"wide range"、"performance"等,这些词不能有效反映软件的类别,具有干扰作用.而 MySQL 的绝大部分标签都具有明确的含义且反映了该软件某些方面的特征.因此,基于软件标签的分类能够获得更好的结果.

表 5 第 4 行列出了基于软件描述和软件标签简单合并的实验结果.实验结果显示,与仅使用软件标签相比,基于该数据能够取得更高的正确率但是其召回率会下降,F_1 值则相近.经过深入分析,合并之所以未能进一步提高分类效果的主要原因在于数据合并方式.这种简单合并将软件标签与软件描述中的词同等对待.但是,由于软件描述的词数通常要远多于软件标签,这种合并虽然能够丰富该软件的信息,但同时也引入了更多的噪声,从而使得软件标签对软件类别的影响减弱,因此没能获得比软件标签更好的效果.

5.2 基于在线属性加权聚合的软件分类

在 3.2.2 节中我们提出了加权聚合的方法,根据软件描述与软件标签的比例对标签进行复制,然后聚合以提高软件标签在合并文本中的权重.在式(1)中,我们设计了参数 α 来控制整体加权强度.本节我们利用 SVM 算法作为分类器进行实验,分析加权聚合方法的有效性以及加权强度对实验结果的影响.我们将 α 的取值从 0 开始不断增加,在[0,2.0]之间以 0.2 为间隔递增,在[2,3]之间每次增加 0.5,共取 13 个不同的 α 值进行实验.其中,α 为 0 时回归为 5.1 节实验中的直接合并.随着 α 的不断增加,软件标签的权重不断增加而软件描述中词的权重不断降低,最终的实验结果如图 4 所示.

从图 4 可以看出,在[0,1]区间,随着 α 的不断增加,分类正确率、召回率以及 F_1 值都在不断增加.其中,F_1 值从 0 时的 62.41%增加到 68.38%.这个结果验证了我们在 5.1 节的分析,说明通过增加标签权重的加权聚合方法既能利用软件描述来丰富软件信息,同时又能够发挥标签质量较好的优点,从而

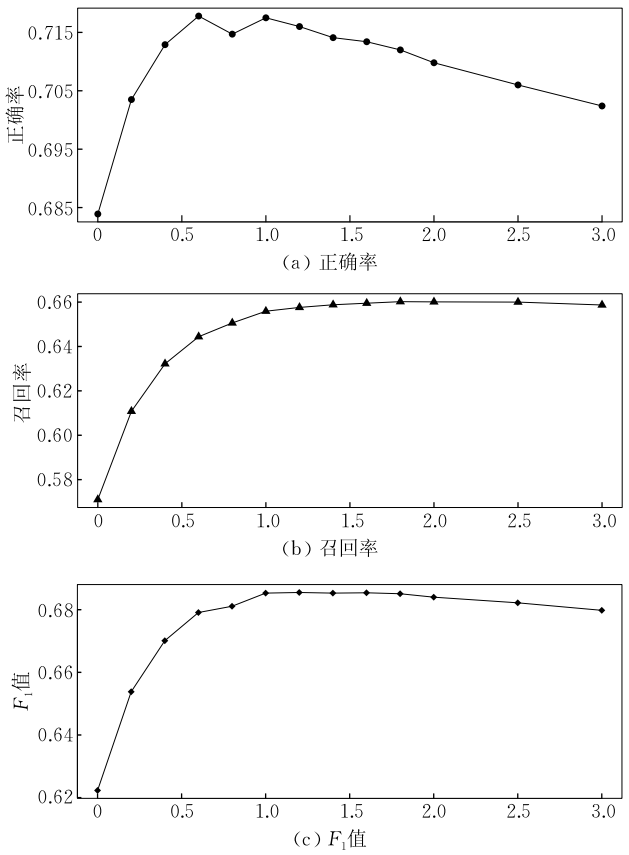


图 4　不同 α 对分类结果的影响

获得更好的分类效果. 此外，从图中可以看出，当 α 从 1 增加到 3 时，分类准确度降低，召回率和 F_1 值基本保持不变. 这主要是因为随着 α 的增大，软件标签的权重会远大于软件描述的权重，使得软件描述在区分软件类别方面的影响越来越小，没有发挥出软件描述中关于软件类别信息的作用.

从 5.1 节和 5.2 节的实验可以看出，软件标签在进行软件分类时比软件描述具有更好的效果. 尽管如此，两者具有一定的互补性，通过加权聚合能够较好地发挥两者的作用，取得更好的分类效果.

5.3　基于在线属性加权聚合和基于 API 的软件分类

基于源代码和字节码分析的软件分类方法被广泛研究，其中基于 API 调用信息的软件自动分类方法根据调用的 API 包/类的类别来对软件类别进行预测被证明是一个有效的方法. Linares-Vsquez 等人[13]针对 Java 程序，通过分析程序中的 API 调用信息对软件在 22 个类别上进行测试. 作者通过对一组开源软件和一组商业软件上进行实验发现基于 API 调用信息的分类方法能够达到和基于源代码标示符/注释的方法近似的效果. 同时，对比实验表明基于 API 包名比 API 方法名分类准确度更高.

本节将本文提出的基于软件在线属性加权聚合的方法与基于 API 包名的分类方法进行对比分析. 我们首先从文献[13]的数据集中抽取出那些在 Ohloh、Freecode 或者 Softpedia 中有标签的项目将其作为测试集，共 849 个软件. 为分析本文方法在该测试集上的效果，我们从 5.1 节实验数据集中将这一部分数据去掉，将剩下的软件作为训练集来构建我们的层次分类器. 本节采用 SVM 方法进行训练和测试，实验参数设置为 $c=0.5$，$\alpha=1$，最终的实验结果见表 6.

表 6　基于软件在线属性和基于 API 的软件分类的结果

类别	F_1值	
	API 调用	在线属性聚合
Bio-Informatics	0.743	0.583
Chat	0.742	0.815
Communications	0.601	0.723
Compilers	0.712	0.616
Database	0.689	0.664
Education	0.633	0.583
Email	0.746	0.792
Frameworks	0.727	0.806
Front-Ends	0.702	0.437
Games/Entertainments	0.728	0.703
Graphics	0.607	0.580
Indexing/Searching	0.732	0.667
Internet	0.604	0.581
Interpreters	0.635	0.679
Mathematics	0.658	0.775
Networking	0.567	0.481
Office/Business	0.598	0.617
Scientific	0.654	0.708
Security	0.622	0.727
Testing	0.730	0.678
Visualization	0.605	0.400
WWW/HTTP	0.696	0.605
平均值	0.6696	0.6464

从表 6 可以看出，两种方法在 22 个类别上的平均 F_1 值相近，其中基于 API 的方法为 66.96%，本文方法为 64.64%. 具体的，在某些类别上，如“Chat”以及“Email”等类别上本文方法取得更好的效果，而基于 API 的方法在其它类别如“Visualization”上更好. 这种差异是由所使用的数据特点决定的. API 包名或方法名与实现相关，通常反映一个软件更细粒度的特征，而软件在线属性则从整体的粒度进行概括. 实验结果表明软件在线属性是一种有效的软件分类源数据. 同时，与 API 数据比较两者从不同的角度和粒度反映软件的特征，在软件分类上具有互补性，对这种互补性的研究将是我们下一步工作的重点.

文献[13]中仅在 22 个粗粒度的类别上进行实验分析，而且没有考虑类别之间的层次关系. 在 SourceForge 中，这些软件实际被标注的类别更多

粒度更细.本文提出的方法根据软件类别之间的关系将其组织为层次结构,并且在较细粒度的类别上也获得了较好的分类效果.例如,在我们构建的分类层次中,“Compiler”和“ Testing”被组织在“Software Development”类别下,“Software Development”还被细分为“Build Tools”、“Quality Assurance”等.测试集上有很多软件同时也属于这些细粒度的类别,本文提出的方法能够将这些软件准确的划分到相应的类别下.表 7 列出了测试软件在“Software Development”下各个类别的测试结果.

表 7 “Software Development”下各子类测试结果

类别	正确率(hP)	召回率(hR)	F_1值(hF)
Software Development	0.8526	0.7826	0.8161
Object Oriented	0.9333	0.6667	0.7778
Build Tools	1.0000	0.5833	0.7368
Algorithms	0.9000	0.4737	0.6207
Quality Assurance	0.8000	0.6667	0.7273
User Interface	0.5484	0.7083	0.6182

5.4 分类方法伸缩性分析

基于互联网的软件开发依赖于互联网软件资源库,而充分利用资源库中的资源依赖于对其中软件的高效分类.本节就本文方法对互联网规模软件资源库进行高效分类的能力进行分析.

本文提出的软件分类方法主要分为训练数据的获取与预处理、分类模型的训练与预测两部分.第一部分,我们设计实现了多线程爬虫程序,启动 50 个线程并行爬取软件主页然后利用正则表达式对 HTML 文件进行解析.我们将程序部署在一个 100 Mbps 带宽接入互联网的服务器上(服务器配置为 8×2.13G CPUs, 16GB RAM 和 2TB 存储).对 Ohloh 中 417344 个软件项目的主页进行爬取和数据抽取的总时间约为 3 天,抽取出的软件在线属性数据大小约为 100 MB.与基于源代码的软件分类相比,本文方法大大降低了数据获取与预处理的代价.软件源代码通常非常复杂,以 MySQL 为例,在 2011 年 6 月,MySQL 源代码中共包括 22 种不同的编程语言,代码文件共包括 1 333 855 行代码以及 298918 行注释.这仅仅是资源库海量软件中的一个中等规模的开源软件,要对资源库中的海量软件源代码进行解析和处理从而进行分类其代价远远超过对软件主页的分析.

分类模型的训练和预测相对于数据获取与预处理而言是非常高效的.该部分的时间开销与所使用的分类算法相关.本文基于 SVM 方法在一台配置为 Intel(R) Core(TM) i5-3320M CPU @ 2.6M, 4GB RAM 的电脑上对 18032 个项目进行 5 倍交叉验证,总时间约为 620 s,其中模型训练时间约为 540 s,分类预测时间为 80 s.基于该数据我们可以算出,在完成分类模型训练后,对 Ohloh 中所有 417344 个软件进行分类的总时间约为 31 min.

在整个层次分类模型构建与预测过程中需要人工参与的部分主要是数据的爬取以及分类层次构建.在图 1 所示的整体流程图中,在 Web 数据爬取与预处理阶段,需要人工确定每个社区中开源项目的 URL 以便爬虫程序爬取项目主页,然后通过分析项目 HTML 页面结构,构建项目在线属性抽取规则.这一部分的工作中,同一个开源社区中项目的 URL 和主页通常具有相同的结构,因此对每个社区只需要人工分析几个项目即可确定.在完成主页爬取之后进行预处理时不需要人工干预,程序能够自动完成相应处理.

本文的分类层次是人工构建的.在分类层次构建时,我们需要对 SourceForge 社区中的分类体系进行统计分析,计算不同类别下的样本数以及类别间关系以确定保留和剔除的类别,构建分类层次.在进行分类预测时,只要将待预测项目的描述及标签作为输入,系统将首先进行自动化预处理、权重计算并将结果作为分类模型的输入,然后根据分类模型计算出该项目可能的类别.

基于上述对本文方法与基于源代码分类方法的对比以及对人工成本的分析可以看出,本文提出的基于软件在线属性的层次分类方法在完成模型构建与部署后能够实现对互联网规模软件资源库中的海量软件的高效分类.

6 结束语

互联网软件资源库需要对海量软件资源进行有效组织和管理.本文构建了一个包括 120 多个类别的层次分类体系,提出了一种基于软件在线属性的层次分类方法,能够实现对海量软件的高效自动分类.该方法基于软件在线属性聚合,不依赖软件源代码且与编程语言无关,因此具有较好的分类效率和伸缩性.我们实现了一个开源软件搜索与分析平台 Influx,搜集了多个大型资源库的开源软件,提供跨资源库项目整合、软件引用分析等功能.针对本文提出的层次分类方法,我们在该平台上实现了相应的 Demo,可以进行访问①.

① http://trustie.influx.net

下一步工作中，我们将获取和分析更多的软件在线属性以进一步提高软件分类的准确度.同时，软件源代码中的标识符以及 API 调用信息从不同的角度和粒度反映软件的功能或技术特征，与软件在线数据具有互补性，未来我们将对多种不同数据如何有效融合以实现更准确高效的分类展开研究.本文重点关注大规模资源库软件资源的组织管理，后续工作中我们将对如何充分利用这些软件资源辅助软件开发展开深入研究.

参　考　文　献

[1] McMillan C. Searching, selecting, and synthesizing source code//Proceedings of the 33rd International Conference on Software Engineering. Hawaii, USA, 2011: 1124-1125

[2] Dumitru H, Gibiec M, Hariri N, et al. On-demand feature recommendations derived from mining public product descriptions//Proceedings of the 33rd International Conference on in Software Engineering. Hawaii, USA, 2011: 181-190

[3] Kobayashi K, Kamimura M, Kato K, et al. Feature-gathering dependency-based software clustering using dedication and modularity//Proceedings of the 28th IEEE International Conference on Software Maintenance. Trento, Italy, 2012: 462-471

[4] Teyton C, Falleri J-R, Blanc X. Mining library migration graphs//Proceedings of the 19th Working Conference on in Reverse Engineering. Ontario, Canada, 2012: 289-298

[5] Zimmermann T, Nagappan N, Gall H, et al. Cross-project defect prediction: A large scale experiment on data vs. domain vs. process//Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'09). Amsterdam, the Netherlands, 2009: 91-100

[6] Surian D, Liu N, Lo D, et al. Recommending people in developers' collaboration network//Proceedings of the 18th Working Conference on Reverse Engineering (WCRE). Limerick, Ireland, 2011: 379-388

[7] Kuang D, Li X, Ling C X. A new search engine integrating hierarchical browsing and keyword search//Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence-Volume Volume Three. Catalonia, Spain, 2011: 2464-2469

[8] Wang T, Yin G, Li X, Wang H. Labeled topic detection of open source software from mining mass textual project profiles//Proceedings of the ACM SIGKDD Workshop on Software Mining (SoftwareMining'12). Beijing, China, 2012: 17-24

[9] Kawaguchi S, Garg P, Matsushita M, Inoue K. Mudablue: An automatic categorization system for open source repositories. Journal of Systems and Software, 2006, 79(7): 939-953

[10] Tian K, Revelle M, Poshyvanyk D. Using latent Dirichlet allocation for automatic categorization of software//Proceedings of the 6th IEEE International Working Conference on Mining Software Repositories. Vancouver, Canada, 2009: 163-166

[11] Ugurel S, Krovetz R, Giles C L. What's the code? Automatic classification of source code archives//Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Alberta, Canada, 2002: 632-638

[12] McMillan C, Linares-Vasquez M, Poshyvanyk D, Grechanik M. Categorizing software applications for maintenance//Proceedings of the 27th IEEE International Conference on Software Maintenance (ICSM'11). Williamsburg, USA, 2011: 343-352

[13] Linares-Vsquez M, McMillan C, Poshyvanyk D, Grechanik M. On using machine learning to automatically classify software applications into domain categories. Empirical Software Engineering, 2012: 1-37

[14] Yu Y, Wang H, Yin G, et al. HESA: The construction and evaluation of hierarchical software feature repository//Proceedings of the 25th International Conference on Software Engineering and Knowledge Engineering (SEKE 2013). Boston, USA, 2013: 624-631

[15] McMillan C, Hariri N, Poshyvanyk D, et al. Recommending source code for use in rapid software prototypes//Proceedings of the 2012 International Conference on Software Engineering (ICSE 2012). Zurich, Switzerland, 2012: 848-858

[16] Li X, Wang H, Yin G, et al. Inducing taxonomy from tags: An agglomerative hierarchical clustering framework//Advanced Data Mining and Applications. Berlin Heidelberg: Springer, 2012: 64-77

[17] Wang S, Lo D, Jiang L. Inferring semantically related software terms and their taxonomy by leveraging collaborative tagging//Proceedings of the 28th IEEE International Conference on Software Maintenance (ICSM). Trento, Italy, 2012: 604-607

[18] Thung F, Lo D, Jiang L. Detecting similar applications with collaborative tagging//Proceedings of the 28th IEEE International Conference on Software Maintenance (ICSM). Trento, Italy, 2012: 600-603

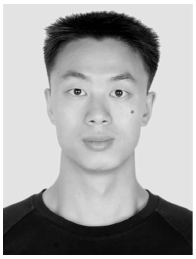
[19] Silla J, Carlos N, Freitas A. A survey of hierarchical classification across different application domains. Data Mining and Knowledge Discovery, 2011, 22: 31-72

[20] Xue G-R, Xing D, Yang Q, Yu Y. Deep classification in large scale text hierarchies//Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'08). Singapore, 2008: 619- 626

[21] Lucca G A D, Penta M D, Gradara S. An approach to classify software maintenance requests//Proceedings of the International Conference on Software Maintenance. Quebec, Canada, 2002: 93-102

[22] Pedregosa F, et al. Scikit-learn: Machine learning in python. Journal of Machine Learning Research, 2011, 12: 2825-2830

[23] Fan R-E, Chang K-W, Hsieh C-J, et al. LIBLINEAR: A library for large linear classification. Journal of Machine Learning Research, 2008, 9: 1871-1874



WANG Tao, born in 1984, Ph. D. candidate. His research interests include software engineering and data mining.

WANG Huai-Min, born in 1962, professor, Ph. D. supervisor. His research interests include software engineering, virtual computing and middleware.

YIN Gang, born in 1975, Ph. D., associate professor. His research interests include software engineering and data mining.

LI Xiang, born in 1988, Ph. D. candidate. His research interests include software engineering and data mining.

YANG Cheng, born in 1991, M. S. candidate. His research interests include software engineering and data mining.

ZOU Peng, born in 1957, professor, Ph. D. supervisor. His research interests include network & information security and distributed computing.

Background

As the rapid growing of the publicly available open source software accumulated in the Internet-scale software repositories, software development is relying more and more on these open source resources. The software developers resort to these resources for searching solutions, learning best practices and so on. However, how to categorize the massive amounts of software in these Internet-scale repositories efficiently to facilitate the utilization of these resources is quite a challenge problem. Many previous works have been conducted on automatic software categorization by analyzing source code or byte code. Most of these works extract the identifiers and comments from the source code and apply different text classification approaches to do categorization. They are language-dependent and only verified on relatively small collections of software projects with flat and coarse-grained categories. Considering the huge amounts of software in these repositories, we need more fine-grained and structured categorization. Besides, as the source code is quite complexity, it is not feasible to categorize the massive amounts of software in the repositories. In this paper, we explore the new software information of online attributes (including descriptions, categorizes and tags) for categorization. To the best of our knowledge, such attributes are less studied in previous works. We propose a hierarchical categorization framework to classify the massive number of software projects hierarchically by using aggregation of different types of online attributes. As such online attributes are natural language and easy to attain and analysis, the categorization based on such information is scalable for Internet-scale repositories. Extensive experiments on more than 18 032 software projects prove the validity and efficiency of our approach.

This work is supported by the National High Technology Research and Development Program of China (Grant No. 2012AA011201) and the National Natural Science Foundation of China (Grant No. 60903043). These projects mainly aim at studying and guiding the trustworthy and large-scale software development in the age of Internet. Our team has made several achievements in this field including several papers in peer review conferences like KDD2012 workshops, ADMA2012, SEKE2013 and ICSM2013. This paper is a quite important part for utilization software resources over the Internet.