AI Bug 分类

刘相君 陈金池 余森垚 黄伟祥

问题定义

在软件从开发到维护的发展历程中，当遇到错误和问题时会将其存档记录，并将其分配给开发人员进行修复。开发人员通常是某特定领域的专家，将特定类型的BUG分配给相关领域的开发人员可以提高软件开发及维护的效率。然而，限于种种现实原因，仅依靠BUG的报告者或开发人员进行对BUG的人工分类难以保证其分类的效率和准确性，我们期望能够自动化地完成对BUG分类的任务。本次软件分析测试大作业，我组将对BUG报告进行数据挖掘，使用有监督学习技术，训练不同的分类模型并进行比较，最终实现一个自动化的，可高效准确地对BUG分类的工具。

研究现状

在这一部分中，我们将讨论软件工程中与漏洞报告分类有关的一些代表性工作。

漏洞报告分类是一个热门的研究领域，Antoniol等人使用ADTree、朴素贝叶斯和逻辑回归等方法对用户报告的文本进行二元分类，判断报告的内容是否为漏洞[1]。使用机器学习算法对漏洞报告进行分类可能会受到数据噪声的影响，Kim等人聚焦于缺陷数据中的噪声检测、噪声消除方法[2]。Dommati等人主要关注漏洞报告分类中的特征提取、减少数据噪声等问题，提出了一种基于朴素贝叶斯的的分类方法[3]。

除了对用户报告做二元分类以判断是否包含漏洞，一些工作还对漏洞的类型进行更细粒度的分类。Limsettho等人使用无监督的方法，通过聚类将漏洞报告自动分组，然后再自动化地为每一组生成一个代表性的名称[4]。Thung等人使用有监督学习的方法，基于SVM将漏洞分为三类，即控制和数据流（control and data flow）, 结构的 （structural）和非功能的（non-functional）[5].

Herzig等人手动检查了来自5个开源项目的7000多个漏洞报告，发现其中33.8%被错误地分类，39%被标记为有缺陷的文件其实并不包含漏洞[6]。一些被误分类的报告并不涉及对代码的修复，而与新功能、文档更新、重构等相关，这样的误分类会使易错代码定位等工作产生偏差。为了应对这一问题，Kochhar等人提出了一种基于多种特征判断漏洞报告是否需要重分类的方法，并对来源于HTTPClient, Jackrabbit, Lucene-Java, Rhino, and 和Tomcat5等五个开源项目的漏洞报告进行预测[7]。Xuan等人则基于朴素贝叶斯和最大期望算法提出了一种半监督的漏洞文本分类方法，以缓解人为标记的训练集中存在的误分类问题[8]。

实验设计

在本次实验中，我们预计分以下几个步骤进行：

1. 查找bug数据，并进行初步清洗过滤
2. 对得到的部分数据进行人工分类，作为后续分类算法的训练、验证集
3. 用不同的分类算法进行训练，并对得到的结果进行比较
4. 最终实验一个可以对bug文本进行自动分类的工具

在第一步，我们初步决定从github上获取某一专题下（比如小程序），所有项目的所有issue，作为实验数据供后续使用。但是issue中有很大的内容是与bug无关的，比如对新功能的请求、对项目的单纯抱怨或者赞美，这些内容在这一步我们是要想办法过滤掉的。此外，在bug报告中还有很多无用的信息，比如粘贴的大段报错代码、项目对issue的规范要求等，这些也是需要去掉的，以防对后续实验的结果产生影响。

在对数据进行人工标记时，我们决定将bug分为9类：Conceptual error、Arithmetic bug、Logic bug、Syntax bug、Resource bug、Multi-threading programming bug、Interfacing bug、Performance bug、Teamworking bug。之后，我们将使用不同的分类算法对得到的标记数据进行建模验证，并对得到的结果进行比较，比较不同算法的各项指标，速度、资源占用、准确度等，最终，我们将挑选出几个效果比较好的算法，将它们封装在一个bug自动分类工具中，使用者可以根据自己的需要选择具有不同特性的算法。最后，我们或许还可以在工具中加入其他的功能，比如对结果进行聚类排序、收集使用者的反馈来调整训练集等，以优化工具的准确性和可用性。

时间节点（共6周）

1. 查找bug数据，并进行初步清洗过滤（2周）
2. 对得到的部分数据进行人工分类，作为后续分类算法的训练、验证集（1周）
3. 用不同的分类算法进行训练，并对得到的结果进行比较（1.5周）
4. 最终实验一个可以对bug文本进行自动分类的工具（1.5周）

实验安排

1. 数据集

我们尝试从GitHub项目中挖掘issues以收集错误。 Issues是程序中错误的自然语言描述，由开发人员编写。 在实验中，我们使用的项目主要是微信小程序。我们计划从1,000个项目中收集错误，执行数据清理并将错误手动分类为不同的类别。

2. 研究问题

- RQ1: 程序中的bug可以被划分为哪些类?

  我们将bug人为地划分为9个类别，并将其中具有无效语义的bugs标记为0，具体类别及详细情况如下所示：

  - 概念错误：这些错误是开发人员对软件必须执行的操作的误解。
  - 算术错误：一些数值计算错误，例如被零除、算术上溢或下溢，以及由于舍入或数值不稳定的

算法而导致的算术精度损失。
    - 逻辑错误：逻辑错误是一种使应用程序无法正常工作的错误，它不会导致崩溃。但是这种错误很难找到。
    - 语法错误：语法错误是应用程序的系统代码中的错误。这是一个很小的语法错误，可以用一个符号表示。编译代码时，编译器会提供有关此类错误的信息，开发人员可以快速修复它们。
    - 资源错误：资源错误是一种常见的类型。例如，空指针取消引用、访问冲突、资源泄漏、缓冲区溢出等。
    - 多线程编程错误：死锁、竞争条件和并发错误都是多线程错误。当程序具有多个同时执行的组件时，可能会出现竞争条件错误。
    - 接口错误：此类错误始终是由API使用不正确、协议实施、硬件处理或其他一些相关错误引起的。
    - 性能错误：性能错误是会导致性能下降并导致不良的用户体验和较低的系统吞吐量的编程错误。
    - 团队合作错误：程序员在开发过程中犯的一些错误，例如未经传播的更新以及文档和产品之间的差异。
- RQ2: 文中使用了哪些分类算法？这些算法分别具有哪些特点?

  我们比较了四种现有的使用度较高的分类算法，分别是k近邻算法(k-nearest neighbor)、决策树分类算法(Decision trees)、朴素贝叶斯分类器(Naive Bayes classifier)、支持向量机算法(Support vector machines)。

    - k近邻算法(k-nearest neighbor)：k近邻是可用于分类问题的算法， 它基本上存储所有可用案例，以其k个邻居的多数票对新案例进行分类。
    - 决策树分类算法：它具有帮助实现分类因变量和连续因变量的通用功能，是一种主要用于分类问题的监督学习算法。 该算法的作用是根据最重要的属性将总体分为两个或更多的同类集合，从而使各组尽可能不同。
    - 朴素贝叶斯分类器：它是一系列以假设特征之间强（朴素）独立下运用贝叶斯定理为基础的简单概率分类器。
    - 支持向量机算法：在此算法中，将每个数据项绘制为n维空间（其中n的值是特征的数量）中的一个点，每个特征的值是特定坐标的值。

3. 预期结果

我们比较了四种不同算法的分类效果，评估标准包括均方误差(MSE)、平均准确率(Mean Accuracy)、R Squared(衡量分类算法的准确率)，我们预期的最佳分类模型的准确率能超过98%。

Reference

[1]Antoniol G, Ayari K, Di Penta M, et al. Is it a bug or an enhancement? A text-based approach to classify change requests[C]//Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds. 2008: 304-318.

[2]Kim S, Zhang H, Wu R, et al. Dealing with noise in defect prediction[C]//2011 33rd International Conference on Software Engineering (ICSE). IEEE, 2011: 481-490.

[3]Dommati S J, Agrawal R, Kamath S S. Bug Classification: Feature Extraction and Comparison of Event Model using Na" ive Bayes Approach[J]. arXiv preprint arXiv:1304.1677, 2013.

[4]Limsettho N, Hata H, Monden A, et al. Automatic unsupervised bug report categorization[C]//2014 6th International Workshop on Empirical Software Engineering in Practice. IEEE, 2014: 7-12.

[5]Thung F, Lo D, Jiang L. Automatic defect categorization[C]//2012 19th Working Conference on Reverse Engineering. IEEE, 2012: 205-214.

[6]Herzig K, Just S, Zeller A. It's not a bug, it's a feature: how misclassification impacts bug prediction[C]//2013 35th International Conference on Software Engineering (ICSE). IEEE, 2013: 392-401.

[7]Kochhar P S, Thung F, Lo D. Automatic fine-grained issue report reclassification[C]//2014 19th International Conference on Engineering of Complex Computer Systems. IEEE, 2014: 126-135.

[8]Xuan J, Jiang H, Ren Z, et al. Automatic bug triage using semi-supervised text classification[J]. arXiv preprint arXiv:1704.04769, 2017.