Software Engineering and Applications 软件工程与应用, 2020, 9(2), 116-123
Published Online April 2020 in Hans. http://www.hanspub.org/journal/sea
https://doi.org/10.12677/sea.2020.92014

Hans汉斯

Multi-Source Software Defect Prediction Model Based on Semi-Supervised Learning

Longhai Yu, Xiaoling Wu*, Jie Lin, Zunhong Xu

Guangdong University of Technology, Guangzhou Guangdong
Email: *11009599@qq.com, 1520978793@qq.com

Received: Mar. 27th, 2020; accepted: Apr. 14th, 2020; published: Apr. 21st, 2020

Abstract

This paper studies the method of establishing a general software defect prediction model between different software projects. By analyzing the project information of multi-source software, this paper designs 25-dimensional software features for machine learning. In order to overcome the differences between different software projects and achieve the generality of the model, a Self-training algorithm based on semi-supervised learning is used to generate a classifier. Finally, the 25-dimensional data features are used to build training data, and a general multi-source software defect prediction model is generated by the Self-training algorithm.

Keywords

Self-Training, Software Defect Prediction, Multi-Source Software

基于半监督学习的多源软件缺陷预测模型

于龙海，吴晓鸽*，凌　捷，许遵鸿

广东工业大学，广东 广州
Email: *11009599@qq.com, 1520978793@qq.com

收稿日期：2020年3月27日；录用日期：2020年4月14日；发布日期：2020年4月21日

摘　要

本文研究在不同的软件项目之间，建立通用软件缺陷预测模型的方法。通过分析多源软件的项目信息，本文设计了25维软件特征用于机器学习。为了克服不同软件项目之间的代码区别，实现模型的通用性，使用基于半监督学习Self-training自训练算法生成分类器。最后利用本文设计的25维数据特征建立训练

*通讯作者。

文章引用: 于龙海, 吴晓鸽, 凌捷, 许遵鸿. 基于半监督学习的多源软件缺陷预测模型[J]. 软件工程与应用, 2020, 9(2): 116-123. DOI: 10.12677/sea.2020.92014

数据，通过Self-training算法生成通用的多源软件缺陷预测模型。

关键词

Self-Training，软件缺陷预测，多源软件

1. 引言

随着计算机软件工程的技术发展，软件项目的代码总量、开源框架、应用场景在日渐增多，软件项目实施过程中使用的开发语言、资源包、开发平台的种类也越来越多。在这种背景下，对软件缺陷预测技术的研究有利于缩短软件项目开发周期，降低项目开发与测试阶段的成本开销。软件缺陷预测帮助项目实施人员将更多的精力集中在项目开发周期中存在潜在缺陷的程序模块，对预测的这些程序模块实施严格的黑盒测试与白盒测试，提升软件项目质量。软件缺陷预测技术要综合考虑项目中功能模块源代码、代码段功能注释、功能设计文档等资源特点，对软件项目中通用特征进行分析与建模，使用缺陷信息对模型进行迭代修正，最后使用预测模型预测不同软件项目中的软件缺陷。

通过对现有学术成果的研究，本文提出了多源软件项目间建立通用软件缺陷预测模型的方法。数据提取方面考虑了项目功能设计文档、源代码特征、代码功能注释等资源。同时分析了项目软件历史仓库中时间维度的信息和开发人员对于项目交流的电子邮件。在这些信息中设计了 25 维软件特征用于模型的训练。在模型的学习阶段，为实现不同项目软件之间缺陷预测模型的通用性，使用基于半监督学习 Self-training 自训练算法生成分类器。该算法无需任何假设，方法简单。最后使用 25 维特征数据训练模型，通过 Self-training 生成通用的软件缺陷预测模型。

2. 相关研究

对于软件缺陷预测的研究已有近 50 年历史。缺陷预测粒度越来越小，缺陷预测模型生成算法从聚类算法逐渐发展到共同学习、迁移学习。对基于机器学习的软件缺陷预测方法，其数据提取阶段中的度量元设计与数据集的预处理、模型生成阶段中缺陷预测模型的构建都对软件缺陷预测模型准确性有不同程度的影响[1]。利用数据引力技术与邻域保留嵌入算法可以构建高效的软件缺陷预测模型[2]。首先在数据提取阶段，利用数据引力技术将目标项目样本与源项目样本相结合，然后再利用典型相关分析使源项目样本与目标项目样本生成公共空间，产生特征相关性最大的训练集，最后利用邻域保留嵌入算法降维得到模型训练集数据。使用生成的样本，经过支持向量机得到项目软件缺陷预测模型。在机器学习的过程中，对训练数据的特征设计、关联性分析尤为重要。利用缺陷训练数据内离散度和支持向量清洗策略，分别从正向增加特征与反向减少特征两个方面进行特征筛选，完成特征的降维。这种方法可以提高模型的运算效率与预测的准确性[3] [4]。利用训练数据中特征的相似度来降维，可以在粗粒度的软件缺陷预测中，降低计算复杂度，提高预测精准度[5]。文献[6]提出 TCS Boost (Transfer Cost-Sensitive Boosting)预测方法。此方法结合源项目与目标项目的部分打标签数据生成训练数据，多次迭代修正模型。基于训练集中的数据构建一个基础分类器，随后逐渐加入训练数据迭代学习，最后生成训练模型。预测软件的稳定

性也非常重要[7]。稳定性预测可以应用于安全软件开发周期的不同开发阶段，并且可以作为软件项目的质量指示器。通过对不同项目的特征提取与降维，文献[8]中提出了跨项目软件缺陷预测框架。此框架可以完成不同软件项目的高效缺陷预测。文献[9]中，提出了基于随机森林算法的半监督学习软件缺陷预测方法。在特征设计阶段，设计了体现变更数量和复杂程度、开发阶段、开发人员数量等方面的 19 维数据特征。通过随机森林的思想，对训练集中，未标注缺陷信息的数据实例，进行自动标注。然后，将自动标注缺陷信息的数据实例，加入之前的训练集。用扩充过的训练集迭代优化模型，最终得到软件缺陷预测模型。

3. 多源软件缺陷预测模型的生成过程

3.1. 训练数据的特征设计

软件缺陷预测可以直接降低项目测试阶段的工作量，同时减少修复缺陷的成本，最终提高软件质量。本文研究使用机器学习的模型生成方法来建立多源软件的通用缺陷预测模型。模型的性能主要取决于训练数据的特征设计。现阶段的软件项目类型可分为 WEB 类软件、中间件、移动 APP、嵌入式软件、操作系统类软件等，这些软件项目在开发工程中所使用的开发工具包、整体软件架构、编程语言都有各自的特点，而我们要从软件项目的通用性出发，设计训练数据集的特征，最终才可以训练出多源软件的通用缺陷预测模型。

训练集中的数据来自软件历史仓库以及开发人员在项目开发过程中用于协同办公的电子邮件，首先从中抽取程序模块与软件项目开发设计文档。为了提高最后预测模型的通用性与准确性，要综合考虑软件缺陷发生的阶段与缺陷类型。根据不同软件项目的多个开发阶段的缺陷报告和运维报告，软件缺陷可能发生在项目需求分析阶段、项目软件设计阶段、项目源程序编码及功能单元测试阶段、整体系统测试阶段等。不同开发阶段中软件缺陷出现的概率也会随着开发阶段推进而逐渐增加。不同种类的缺陷会分布在项目软件的各个模块。对于大型软件项目来，系统中会存在需求设计缺陷、功能设计缺陷、编码缺陷、性能缺陷、可靠性缺陷和安全性缺陷等。

一个软件项目遵循标准的安全软件开发周期，其项目源代码不断迭代，直至项目完成。本文对于项目信息，以每次的项目代码变更作为项目状态提取周期。在初始代码未变更前，我们认为项目是安全的，即初始状态项目为空，软件缺陷数量是 0。软件缺陷全部出现在每次代码变更的过程中。提取特征时，都以一次变更结束后到下一次变更完成期间为周期。

首先设计每次变更对软件项目中代码总量影响的特征。不同类型项目之间，文件目录结构和所使用的开发软件包会有明显的区别，但是在代码总量方面，软件项目的代码总量越多，表明此软件项目越复杂。变更越复杂的项目，越容易在项目中引入软件缺陷。同时变更的代码量越多，越容易引入软件缺陷。软件项目发生变更时，在代码总量方面设计的特征如表 1 所示。

Table 1. Features of total code
表 1. 代码总量方面的特征

序号	特征名称	描述	选取动机
1	CLA/CLD	变更所增加或减少的代码行数量	在变更中，增加或减少的代码行数越多，越有可能引入软件缺陷
2	FQD	变更所影响的项目文件在变更发生前的代码行数量	变更的项目文件越大，那么此次变更越有可能在该文件中引入软件缺陷
3	PSC	变更所影响的项目文件中，代码注释行数与代码总行数的比值	该比值越大，说明变更的文件越复杂，越有可能引入软件缺陷

项目文件中，子文件数目与代码目录数目也是体现软件项目复杂性的重要特征。而越复杂的项目，存在软件缺陷的可能性越大。对软件项目的变更，所影响的子文件数目越多，那么此次变更更容易引入软件缺陷，当软件项目发生变更时，在项目子文件与目录方面设计的特征如表 2 所示。

Table 2. Features of project subfiles
表 2. 项目子文件与目录的特征

序号	特征名称	描述	选取动机
4	NSD	变更所影响的子文件数量	变更影响的子文件越多，越有可能引入软件缺陷
5	NCD	变更所影响的代码目录数量	变更影响的代码目录越多，越有可能引入软件缺陷
6	NFD	变更的项目文件在变更发生前，子文件数量	变更的项目中子文件越多，说明变更的项目越复杂。那么此次变更越有可能在该文件中引入软件缺陷
7	NCE	变更的代码在变更相关文件中的分布(用信息熵计算)	信息熵越大，变更的代码在文件中越分散。那么对于项目开发人员的挑战越大，越有可能引入软件缺陷
8	FTF	变更所影响的子文件中，变更修改的代码数量	变更的子文件中，代码修改的越多，越有可能引入软件缺陷
9	FS	变更所影响的子文件在变更发生前后抽象语法树的差异(使用树距离计算)	变更所引入的抽象语法树变化与引入软件缺陷有关联
10	DS	变更所影响的子文件在变更发生前后目录树的差异(使用树距离计算)	如果变更带来了代码目录的变化，说明此次变更对项目功能进行了修改。修改越多，越有可能引入软件缺陷
11	NDE	对此变更所影响的子文件进行过变更的开发者数量	对同一子文件进行过变更的开发者越多，越有可能对该文件引入软件缺陷
12	NFIX	对非修复缺陷变的更影响的子文件进行过修复的变更数量	文件已知的修复越多，说明该文件越容易出现软件缺陷。那么非修复的变更越容易引入新软件缺陷
13	AGE	此次变更与影响相同子文件的上一次变更之间的时间差	对于一个文件的修改越频繁，越有可能引入软件缺陷

对于软件项目的变更可分为修复缺陷的变更与其它变更两种。修复缺陷的变更会减少软件项目原有的软件缺陷数量，但是往往会很复杂，修改的代码结构会非常多。那么修复缺陷的变更引入新的软件缺陷的概率会更大。且对于提交的变更，对于此变更审查的人数越多，引入软件缺陷的概率越小。对于每次变更的特征设计如表 3 所示。

Table 3. Features of software project changes
表 3. 软件项目变更的特征

序号	特征名称	描述	选取动机
14	FIX	此次变更是否为修复缺陷的变更	修复缺陷的变更会减少项目文件中原有的软件缺陷数量，但是引入新的软件缺陷概率会更大
15	NBR	与此次变更相关的缺陷报告的数量	缺陷报告越多，变更修改的代码越多，更容易引入软件缺陷
16	LTF	与变更相关的日志大小	与变更相关的日志越大，说明此次变更越复杂，引入软件缺陷的可能性越大
17	IN	变更归档前，被修正的次数	在变更归档前，变更修正越多，其引入缺陷的可能性越小
18	RPD	对变更进行审查的人数	对变更审查人数越多，其引入缺陷的可能性越小
19	CPD	对变更的审查意见数量	对变更的审查意见越多，说明该变更经过更多开发人员的审阅，该变更引入软件缺陷的可能性越小
20	RPT	变更被创建到归档之间的时间	变更被审查的时间越短，越有可能引入软件缺陷

本文设计的数据特征充分考虑了软件项目中开发人员的能力。因为软件项目中的源代码，一部分来自开源或闭源的软件开发包，其余源代码及项目功能设计都由开发人员编写。所以开发人员的能力及项目复杂程度直接决定了项目中软件缺陷出现的特征。变更发生时，在项目开发人员方面设计的特征如表 4 所示。

Table 4. Features of project developers
表 4. 项目开发人员的特征

序号	特征名称	描述	选取动机
21	EPD	开发者已提交的变更数量	提交过的变更越多，表明开发者更有经验。则不容易引入软件缺陷
22	REPD	开发者近期提交的变更数量	近期经常提交变更的开发者对此项目更加熟悉，则不容易引入软件缺陷
23	SEP	开发者已提交变更中，影响到此次变更相关子系统的数量	已提交的变更影响此次变更相关子文件的数量越多，说明开发者对相关子系统越熟悉，越不容易引入软件缺陷
24	PTL	开发者参与此次变更相关子系统的时长	在变更相关子系统中开发时间越长，开发者对子系统越熟悉，越不容易引入软件缺陷

整个软件项目开发周期中，后一个开发阶段受前一个开发阶段工作成果影响。往往因为前一个开发阶段引入的软件缺陷，引起后面开发阶段要解决前面开发阶段遗留的问题，并解决此阶段新引入的软件缺陷。在大量的软件项目中，不同的开发阶段出现的软件缺陷呈现规律性。变更发生时，在软件开发周期方面设计的特征如表 5 所示。

Table 5. Features of the software development cycle
表 5. 软件开发周期的特征

序号	特征名称	描述	选取动机
25	LTF	变更提交时，项目所处的开发阶段	不同项目，在同一开发阶段所要完成的目标相近。不同开发阶段与引入软件缺陷的概率有关联

以上便是本文设计的用于半监督学习的 25 维数据特征，其中考虑了项目源码特征、开发阶段、项目目录与文件结构、开发人员能力、变更意见审核等信息。同时加入了软件开发周期、系统开发流程等通用于不同项目的软件特征，以此设计用于多源软件缺陷预测的训练数据特征。

3.2. 模型生成

根据多源软件的通性特征，本文设计了 25 维训练数据特征。在模型学习阶段，选用半监督学习 Self-training 自训练算法，得到最后的多源软件通用软件预测模型。模型生成过程如图 1 所示。

建立模型所需要的原始数据可由开发人员使用的邮箱系统、软件历史仓库中的版本控制系统和缺陷追踪系统提供。经过特征提取，得到 25 维特征的学习训练集与测试集。其中，一部分特征维度的数据不适合计算，对这些数据进行特征量化。本文设计的 25 维数据特征中，绝大部分特征可直接提取特征对应的数据标量，剩余的特征需设计度量方法，如表 6 所示。

完成数据预处理后，随机取出一部分数据，进行人工检查，为每条数据打上标签(表明数据是否有软件缺陷)。将这一小部分人工打过标签的数据，输入到 Self-training 自训练算法中，计算得到初始化分类器 A。之后逐步加入未打标签的数据，进行半监督学习。迭代修正模型的过程中，每次从没有打标签的数据集中，取出一小部分数据子集，加入训练数据中，再次训练分类器 A，重新得到分类器。对于每一

次加入训练的数据选取，满足每一次选取的数据子集加入训练集后，新训练的分类器与前一次训练的分类器神经元输入突触的权值变化尽可能小的条件。以此条件选取数据，逐次迭代训练分类器。最终得到通用的多源软件缺陷预测模型。缺陷预测模型的学习过程如图 2 所示。

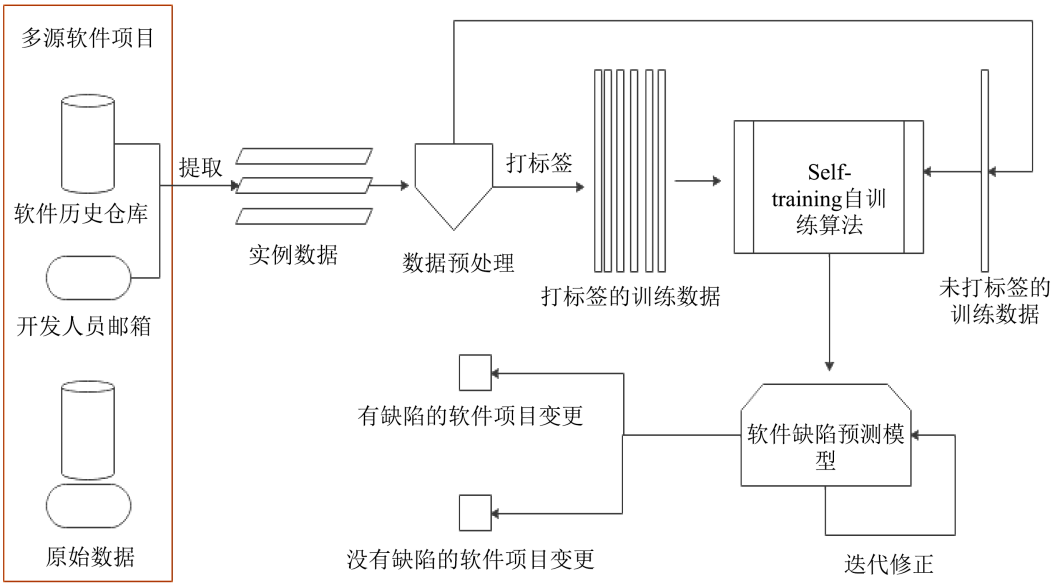

Figure 1. Model generation process
图 1. 模型生成过程

Table 6. Measurement of some features
表 6. 部分特征的度量方法

特征序号	特征名称	预处理量化方法	动机
3	PSC	取所影响文件中，注释行数目与代码总行数的比值，以小数点后保留两位有效数字的小数形式提取该特征。	因为在调节神经元突触权值的过程中，计算机以浮点数存储神经网络数据。取小数形式也与其它属性值在格式上保持一致。
7	NCE	用信息熵计算变更对子文件的影响，此熵值同样采用以小数点后保留两位有效数字的小数形式。	熵值能够反映变更对子文件的影响。与其它属性值在格式上保持一致。
9	FS	取变更发生前后抽象语法树的树距离差值。	此距离差值可以反应变更对子文件的影响。
10	DS	取变更发生前后子文件目录树的树距离差值。	此距离差值可以反应变更对子文件功能结构的影响。
13	AGE	此时间差以分钟为单位，取整数。	统一将所有特征中涉及时间的值都以分钟为单位。取整数形式能够减少后续计算量。
14	FIX	若为修复缺陷的变更，值取 1，否则值取 0。	两种情况分别用 1，0 表示，形式简单。
16	LTF	取与变更相关的日志数量，即日志条数。	取值为整数，方便后续计算。
20	RPT	此时间差以分钟为单位，取整数。	与其它时间特征形式统一。
24	PTL	此属性值以天为单位取值，取整数。	该属性可能横跨整个项目开发周期，避免该值过大，增大后续计算量。
25	LTF	从需求阶段开始，测试阶段结束。依次为需求阶段取值 1、设计阶段取值 2、开发阶段取值 3、测试阶段取值 4。	取小整数，减少后续计算量。

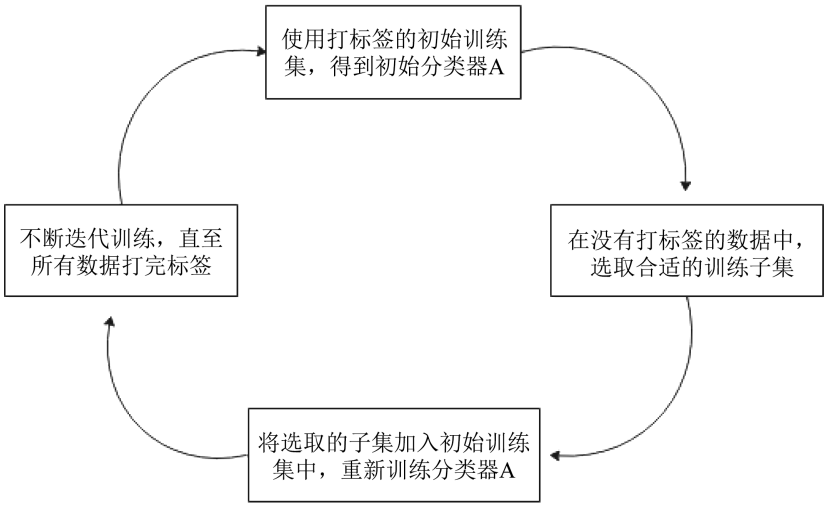

Figure 2. Model training process
图 2. 模型训练过程

得到软件缺陷预测模型后可以通过查准率、查全率来反向评估模型效果。而通过预测结果可以探究软件缺陷数据特征及其分布特点，提高特征数据的提取效率。

与文献[9]中提出的基于随机森林算法的半监督软件缺陷预测方法相比，本文在数据特征设计阶段，充分考虑了项目代码中的代码注释对一次软件项目变更的影响、变更产生的日志与引入软件缺陷相关联的可能性、软件缺陷与抽象语法树相关联的可能性、软件项目中开发者能力与引入软件缺陷的关联性、以及不同类型变更与引入软件缺陷的关联性等。本文设计了软件项目中的代码注释、变更日志、抽象语法树、开发者能力等方面的相关特征，使得训练数据能够更全面地反映出每次软件项目变更的特点。

4. 结束语

本文研究了多源软件通用软件缺陷预测模型的建立方法。通过研究多源软件项目中软件设计文档、软件源代码以及软件代码注释等资源在软件开发周期的作用，通过分析项目软件历史仓库与项目开发人员电子邮件的信息，设计了体现软件项目的 25 维软件缺陷数据特征。为实现建立不同项目软件之间通用的软件缺陷预测模型，使用基于半监督学习 Self-training 自训练算法生成分类器。最后使用 25 维特征数据训练模型，通过 Self-training 生成通用的软件缺陷预测模型。

基金项目

广东省重点领域研发计划项目资助(项目编号 2019B010139002)；广州市科技计划项目资助(项目编号 201902020006、201902020007、201902010034)。

参考文献

[1] 张志武. 基于机器学习的软件缺陷预测方法研究[D]: [博士学位论文]. 南京: 南京邮电大学, 2018.

[2] 黄琳. 基于度量元的静态跨项目软件缺陷预测技术研究[D]: [硕士学位论文]. 南京: 南京邮电大学, 2019.

[3] 李梦奇. 基于机器学习的软件缺陷预测方案研究[D]: [硕士学位论文]. 北京: 北京邮电大学, 2019.

[4] Zhang, F., Hassan, A.E., Mclntosh, S., *et al*. (2017) The Use of Summation to Aggregate Software Metrics Hinders the Performance of Defect Prediction Models. *IEEE Transactions on Software Engineering*, **43**, 476-491. https://doi.org/10.1109/TSE.2016.2599161

[5] Yu, Q., Jiang, S., Wang, R., *et al*. (2017) A Feature Selection Approach Based on a Similarity Measure for Software

Defect Prediction. *Frontiers of Information Technology & Electronic Engineering*, **18**, 1744-1753. https://doi.org/10.1631/FITEE.1601322

[6] Ryu, D., Jang, J. and Baik. J. (2017) A Transfer Cost-Sensitive Boosting Approach for Cross-Project defect Prediction. *Software Quality Journal*, **25**, 235-272. https://doi.org/10.1007/s11219-015-9287-1

[7] Shi, Y., Li, M., Arndt, S., *et al*. (2017) Metric-Based Software Reliability Prediction Approach and Its Applocation. *Emprical Software Engineering*, **22**, 1579-1633. https://doi.org/10.1007/s10664-016-9425-9

[8] Jing, X.Y., Wu, F., Dong, X., *et al*. (2017) An Improved SDA Based Defect Prediction Framework for Both Within-Project and Cross-Project Class-Imbalance Problems. *IEEE Transactions on Software Engineering*, **43**, 321-339. https://doi.org/10.1109/TSE.2016.2597849

[9] 何清. 基于半监督学习和投票决策理论的软件缺陷预测[D]: [硕士学位论文]. 上海: 上海交通大学, 2017.