基于相似度匹配的软件缺陷预测方法研究

刘雅新，吴高艳，何 鹏

（湖北大学 计算机与信息工程学院，湖北 武汉 430062）

摘 要：针对跨项目缺陷预测（Cross－Project Defect Prediction，CPDP）中为目标项目选择合适的训练数据问题，在已有相似度匹配方法的基础上，引入项目情境信息，从而提出一种改进的 CPDP 预测模型。实验结果表明：引入项目的情境信息，有助于提高 CPDP 性能；所提方法的 F-measure 值比已有方法提高了 15.04％和 6.57％，但相比 WPDP 方法，仍有待提高。

关键词：软件质量保证；缺陷预测；相似度匹配；训练数据选择

DOI：10.11907/rjdk.171465

中图分类号：TP301 **文献标识码**：A **文章编号**：1672-7800（2017）008-0009-03

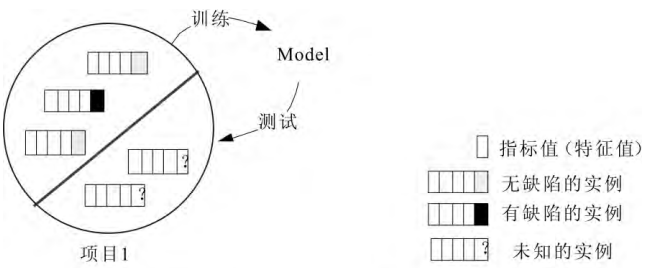
0 引言

软件缺陷预测研究一直是软件工程领域中的热门方向，最早可以追溯到上世纪 70 年代。常规的方法是利用项目自身已有历史数据训练预测模型后，再用于后续版本的缺陷预测，即项目内缺陷预测（Within-Project Defect Prediction，简称 WPDP），如图 1（a）所示。然而，已有研究表明训练高质量的 WPDP 模型要求有充足的历史数据，这对一些新项目或还不活跃的软件项目便难以满足。

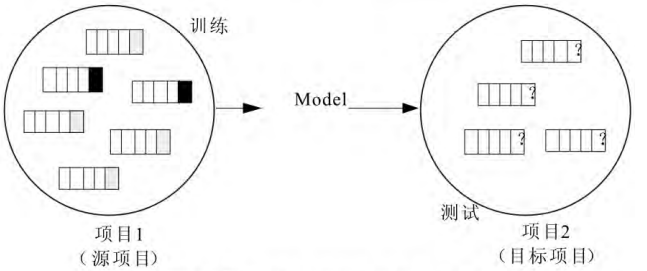
近些年来随着互联网的蓬勃发展，尤其是开源社区如 Github 的兴起，互联网上可供获取的公开缺陷数据集越来越多，而且数量仍在不断增长。为有效利用互联网上已有的丰富数据资源，一些研究者提出利用其它软件项目的数据来训练，构建跨项目的缺陷预测模型（Cross－Project Defect Prediction，简称 CPDP），用于解决 WPDP 中训练数据受限的瓶颈[1-5]，如图 1（b）所示，为软件缺陷预测研究开辟了一条崭新的途径。

在 CPDP 早期研究中，都是将来自其它软件项目的所有数据作为训练集，并不涉及训练集的精简。常常出现因训练数据包含过多噪声，从而降低了模型准确性[5]。在某种程度上，数据的质量远比数量对 CPDP 性能的影响更大。然而，如何才能从大量的可供使用的缺陷数据中挑选出质量更高的部分用于预测模型训练，仍然是 CPDP 研究中急需解决的一个问题[6]。

针对以上问题，目前主要有两种思路。一种是通过特征降维的方法减少冗余指标信息，从而减少数据噪音来改善缺陷预测的性能和效率[7]。另一种方法则是本文将考虑的通过减少数据量来减少重复的无价值的实例[8]。在训练数据总量的精简方面，以往研究都只是根据数据的度量指标信息进行相似度匹配，再返回 top－k 个最相关的实例构成新的训练数据集，但它们并没有充分考虑项目的情境信息。实践中，每个项目的情境信息存在差异，例如项目的主题、服务对象、编程语言等。



（a）项目内缺陷预测（Within-Project Defect Prediction，WPDP）



（b）跨项目缺陷预测（Cross-Project Defect Prediction，CPDP）

图 1 软件缺陷预测模型

本文在为 CPDP 预测选择合适跨项目训练数据集时，通过考虑项目的 5 个常规属性信息（包括项目主题、目标

基金项目：湖北省知识创新专项项目（2016CFB309）；湖北大学自科青年基金项目（201507）

作者简介：何鹏（1988－），男，江西宜春人，博士，湖北大学计算机与信息工程学院讲师，研究方向为软件度量、软件维护、复杂网络。

受众、编程语言、运行环境、开源认证),并利用自然语言处理中的 TF－IDF 技术将它们量化,从而得到每个项目的情境信息向量。最后,结合项目的情境信息与项目中实例特征值计算数据集的相似度。本文的主要贡献可归纳为:

(1)引入项目的情境信息,提出一种改进的基于相似度匹配的 CPDP 预测方法,并使 CPDP 预测性能得到改进。

(2)验证本文方法的 CPDP 预测模型在朴素贝叶斯分类器下效果最好。

1 跨项目缺陷预测(CPDP)

CPDP 形象表示为利用其它项目组成的缺陷数据集 $S=\{P_1,P_2,\cdots,P_s\}$ 对目标项目 P_t 作缺陷预测。假设一个项目 P 由 n 实例(类文件)组成,即 $P=\{I_1,I_2,\cdots,I_i,\cdots,I_n\}$,实例 I_i 表示为 $I_i=\{f_{i1},f_{i2},\cdots,f_{ij},\cdots,f_{im}\}$,$f_{ij}$ 为实例 I_i 在第 j 个度量指标上的值,m 为用于度量实例的指标个数。一个项目数据集 P 中度量指标 F_i 对应的向量可表示为 $F_i=\{f_{1i},f_{2i},\cdots,f_{ji},\cdots,f_{mi}\}$,$f_{ji}$ 为第 j 个实例在该度量指标上的值,各实例指标值的分布特性可表示为 $C_i=\{SC_{i1},SC_{i2},\cdots,SC_{ik}\}$,$SC$ 为对应的度量指标值的分布特性(最大值、最小值、中位数、均值和标准方差)。因此,项目 P 可根据度量指标量化为 $V=\{C_1,C_2,\cdots,C_k,\cdots,C_m\}$。这样,项目 A 和 B 之间的相似性可表示为:

$$Sim(A,B)_{metric}=\cos(V_A,V_B)=\frac{V_A\cdot V_B}{|V_A||V_B|} \quad (1)$$

另外,假定每个项目情境信息按照主题、目标受众、编程语言、运行操作系统、认证顺序来进行量化表示,则可以表示为 $U=(AT_t,AT_{ia},AT_{pl},AT_{os},AT_{lic})$,其中 AT_i 分别为上述属性前面几个字母的缩写。因此,项目 A 和 B 之间的属性相似性可表示为:

$$Sim(A,B)_{context}=\cos(U_A,U_B)=\frac{U_A\cdot U_B}{|U_A||U_B|} \quad (2)$$

而表示项目情境信息的每个属性本身又为一个向量 $AT_i=(w_{i1},w_{i2},\cdots,w_{im})$,$m$ 为属性 i 中包括的元素种类,因此,项目情境信息相似性为所有 5 个属性向量下的余弦相似性总和,可用式(3)表示,系数 α 表示每个属性的比重系数,本文视每个属性具有相同的重要性,即 $\alpha=0.2$。

$$Sim(A,B)_{context}=\sum_{i=1}^{n}\alpha_i\cos(A_{AT_i},B_{AT_i}) \quad (3)$$

对于每个属性向量 AT_i 中的 w_{ij},可通过修改后的 tf,idf 表示为式(4):

$$w_{ij}=(\log f_{ij}+1)\log\frac{\#p}{\#p_j} \quad (4)$$

其中,f_{ij} 为项目在属性 i 元素 j 上的频率,$\#p$ 和 $\#p_j$ 分别代表项目总数和具有元素 j 的项目数。结合式(3)和式(4),得到:

$$\cos(A_{AT_i},B_{AT_i})=\frac{\sum_{j\in A_i\cap B_i}(\log f_{ij}+1)(\log f_{ij}+1)(\log\frac{\#p}{\#p_j})^2}{\sqrt{\sum_{j\in A_i}((\log f_{ij}+1)\cdot\log\frac{\#p}{\#p_j})^2\sum_{j\in B_i}((\log f_{ij}+1)\cdot\log\frac{\#p}{\#p_j})^2}} \quad (5)$$

最后,在跨项目数据集选择过程中,结合项目的两种信息,得到最终的相似度得分:

$$Score(A,B)=Sim(A,B)_{metric}+Sim(A,B)_{context} \quad (6)$$

对应算法实现描述如下:

算法 1:基于相似度匹配的 CPDP

输入:

1. 候选训练数据集 $S=\{P_1,P_2,\cdots,P_t\}$;
2. 目标项目数据集 $P_t=\{I_1,I_2,\cdots,I_i,\cdots,I_{n1}\}$;

输出:

3. 返回预测结果 *result*。

方法实现:

4. P 是为 P_t 从候选集 S 中返回的最相似的数据集;
5. 初始化 $P\leftarrow\Phi$
6. for 每个项目 $P_i(P_i\in S)$ do;
7. // 对 P_i,P 进行相似度匹配
8. Set $tempScore\leftarrow Score(P_i,P_t)$
9. end for
10. //取集合 $tempScore$ 中与目标项目 P_t 最相似的项目
11. $P\leftarrow Max(tempScore)$;
12. //用选择的数据集 P 训练模型并对 P_t 进行预测
13. $result\xleftarrow{P_t}model(P)$;//CPDP 预测
14. 返回 *result*

2 实证分析

2.1 数据集

本次实验使用 PROMISE 提供的 10 个项目缺陷数据集,表 1 给出了每个项目的相关信息。数据集中每个实例表示一个类文件(.java),包括 20 个源代码度量指标用于量化实例,其中 CK 套件 10 个、Martins 指标 2 个、QMOOM 套件 5 个、McCabe's CC 指标 2 个,以及代码行 LOC。

2.2 分类器与评价指标

本文采用朴素贝叶斯分类器(Naïve Bayes)作为本次 CPDP 预测模型训练的分类器,Naïve Bayes 是一个基于条件概率最简单的分类器,其之所以被称之为"朴素"是因为它假设所有特征之间都是相互独立的。数学表示为 $P(X|Y)=\prod_{i=1}^{n}P(x_i|Y)$,$X=\{x_1,x_2,\cdots,x_n\}$ 为一个特征向量,Y 为分类变量。尽管现实中这种独立假设并不完全成立,但朴素贝叶斯已在很多实践研究中得到有效的应用[9]。在预测过程中,给定一个新的实例(类文件),朴素贝叶斯分类器通过计算该类在各个特征值上的条件概率的乘积来评估存在缺陷的概率,式(7)为基本的计算公式:

$$P(Y=k \mid X)=\frac{P(Y=k)\prod_i P(x_i \mid Y=k)}{\sum_j P(Y=k)\prod_i P(x_i \mid Y=k)} \tag{7}$$

表 1　实验数据信息

序号	项目	版本号	缺陷率(%)
1	Ant	1.3	19.6
2	Camel	1.2	18.9
3	Ivy	1.4	24.9
4	Jedit	3.2	28.8
5	Lucene	2.2	54.9
6	POI	2.5	49.8
7	Synapse	1.2	23.6
8	Velocity	1.4	58.5
9	Xalan	2.4	37
10	Xerces	1.1	38

预测过程中存在 4 种可能情况：假阳性(FP)、假阴性(FN)、真阳性(TP)与真阴性(TN)。根据 4 种预测结果可计算准确率(precision)、召回率(recall)和 F-measure 评价指标。

准确率用来衡量有多少真实存在缺陷的实例被成功地预测。准确率越高，表明无缺陷实例被误认为有缺陷的情况会更少。

$$precision=\frac{TP}{TP+FP} \tag{8}$$

召回率用来衡量被预测为有缺陷的实例中有多少是真实存在缺陷。召回率越高，表明有缺陷实例被误认为无缺陷的情况会更少。

$$recall=\frac{TP}{TP+FN} \tag{9}$$

F-measure 用来平衡准确率与召回率，为两者的加权平均值。F-measure 越接近 1 表示预测效果越好。

$$F\text{-}measure=\frac{2*precision*recall}{precision+recall} \tag{10}$$

2.3　实验结果

问题 1：引入项目的情境信息，是否有助于提高 CPDP 性能？

实验结果如图 2 所示，不难发现，在为目标项目选择合适的跨项目数据集过程中，引入项目的情境信息后的 CPDP 模型性能比不使用时整体都有所提高，其中项目 Synapse、Ant 和 Jedit 三个项目性能改进最为显著，F-measure 值改进幅度分别为 0.32(86.7%)、0.222(66.2%)和 0.207(64.3%)。因此，实验结果证实，引入项目的情境信息，有助于提高 CPDP 性能。

问题 2：相比已有 CPDP 方法，本文所提方法是否性能更好？

为了进一步验证本文所提出的方法的有效性，引入文献[1]、[5]中提出的 CPDP 方法作为比较对象。此外，与 WPDP 方法进行对比，如图 3 所示。结果显示，本文所提方法相比两种基准方法 baseline1 和 baseline2 整体性能分布有所提高，表现为前者均值为 0.512，两种基础方法的均值分别为 0.445 和 0.481，改进比例分别为 15.04%和 6.57%，且最大值和最小值均有所提高。然而，本文方法相比于 WPDP 方法，仍然表现出一定差距。WPDP 的均值为 0.621，说明选用其它项目的数据训练出的模型仍不如目标项目自身的数据更可靠。尽管如此，但考虑到现实中，对于一些新项目或不活跃的项目，它们可供使用的历史数据并不多，在此数据不充分的情境下，即便是有 WP-DP 方法，相信效果也依旧不佳。

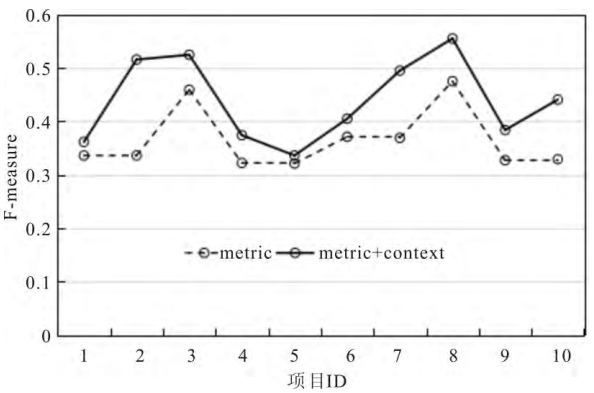


图 2　引入情境信息后的 CPDP 方法性能

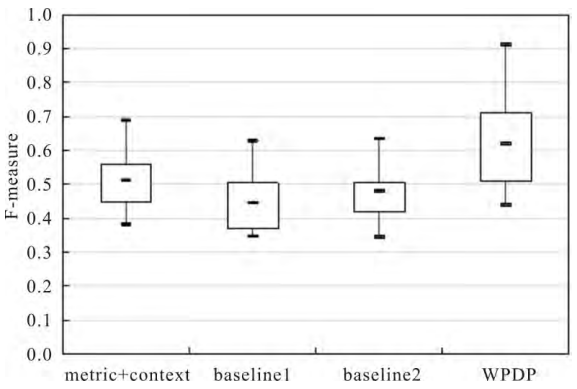


图 3　对比结果

本文不足之处：①实验数据只选取了 10 个项目，实验结论还有待在更多项目数据集上加以验证；②本文只考虑了 5 个项目属性用于表达情境信息，根据开源社区中提供的信息，表达项目情境的属性还有很多，有待进一步探索。

3　结语

本文围绕跨项目缺陷预测开展研究，针对为目标项目选择合适的训练数据问题，在以往通过项目实例度量指标相似度匹配的基础上，引入项目的情境信息，从而改进 CPDP 预测模型。实验结果表明：①引入项目的情境信息，有助于提高 CPDP 性能；②相比两种基准方法，笔者的方法整体性能有提高，分别提高 15.04%和 6.57%，但相比 WPDP 方法，依旧还有待提高。

参考文献：

[1] HE Z, SHU F, YANG Y, et al. An investigation on the feasibility of cross-project defect prediction[J]. Automated Software Engineering, 2012, 19(2):167-199.

(下转第 14 页)

本文实验中值取 200。

表 1 不同 top k 分析时间意图方法的检索性能

方法	MAP	P@5	P@10	ndcg@20	err@20
LMU-DIF(top100,α=0.9)	0.2449	0.5520	0.4820	0.3265	0.1775
LMU-DIF-rankAdd(top100,α=0.9)	0.2446	0.5560	0.4900	0.3264	0.1804
LMU-DIF (top200,α=0.92)	0.2450	0.5640	0.5000	0.3330	0.1829
LM-DIF-rankAdd(top200,α=0.8)	0.2430	0.5700	0.5110	0.3300	0.1839
LMU-DIF (top300,α=0.9)	0.2446	0.5640	0.5060	0.3123	0.1803
LM-DIF-rankAdd(top300,α=0.9)	0.2442	0.5640	0.4980	0.3202	0.1795

语言模型中计算值需要考虑时间不确定性，不同的时间间隔可能会影响排名结果。表 2 列出不同时间间隔(0 天、7 天、30 天、3 个月)下 LMU－DIF 和 LMU－DIF－rankAdd 方法的指标值。从表 2 可见，时间间隔取太大或太小都会降低结果性能，间隔 7 天时性能最好。

表 2 不同时间间隔方法的检索性能

方法	MAP	P@5	P@10	ndcg@20	err@20
LMU-DIF(α=0.9)	0.2408	0.5549	0.4950	0.3291	0.1769
LMU-DIF-rankAdd(α=0.9)	0.2446	0.5629	0.4920	0.3295	0.1832
LMU-DIF(7days,α=0.92)	0.2450	0.5640	0.5000	0.3330	0.1829
LMU-DIF-rankAdd(7days,α=0.8)	0.2430	0.5700	0.5110	0.3310	0.1839
LMU-DIF(30days,α=0.9)	0.2408	0.5588	0.5048	0.3305	0.1763
LMU-DIF-rankAdd(30days,α=0.9)	0.2442	0.5628	0.5020	0.3319	0.1788
LMU-DIF(3months,α=0.9)	0.2406	0.5507	0.4940	0.3273	0.1767
LMU-DIF-rankAdd(3months,α=0.9)	0.2445	0.5625	0.4960	0.3316	0.1797

综合上面的分析，表 3 列出了每个方法在参数配置最优情况下各指标的值，Baseline 是仅考虑内容相关性的一个基准。总体上看，各种方法性能都有所提升，LMU－DIF－rankAdd 方法比 LMU－DIF 更优，但都优于 Baseline，表明本文提出的方法在改善搜索引擎性能方面有一定效果，排名模型需要考虑时间因素的影响。

表 3 最优情况下各方法的检索性能

方法	MAP	P@5	P@10	ndcg@20	err@20
Baseline	0.2409	0.5490	0.4980	0.3299	0.1737
LMU-DIF (α=0.92)	0.2450	0.5640	0.5000	0.3330	0.1829
LMU-DIF-rankAdd(α=0.8)	0.2430	0.5700	0.5110	0.3310	0.1839

4 结语

本文提出一种支持隐式时间查询的文档排名方法，该方法首先分析隐式查询的时间意图，在此基础上线性计算时间相关性得分，结合时间相关性得分和内容相关性得分，把重排结果返回给用户。实验结果表明本方法具有一定的实用价值。

参考文献：

[1] NUNES S, RGIO, RIBEIRO C, et al. Use of temporal expressions in web search, proceedings of the Ir research[C]. European Conference on Advances in Information Retrieval, 2008.

[2] METZLER D, JONES R, PENG F, et al. Improving search relevance for implicitly temporal queries [J]. Proceedings of Sigir', 2009(1):700-701.

[3] ALONSO O, STROTGEN J, BAEZA YATES R, et al. Temporal information retrieval: challenges and opportunities [J]. Temporal Web Analytics Workshop at Www, 2011(1):8-9.

[4] BERBERICH K, BEDATHUR S, ALONSO O, et al. A language modeling approach for temporal information needs [M]. ECIR, 2010.

[5] JONES R, DIAZ F. Temporal profiles of queries [J]. Acm Transactions on Information Systems, 2007, 25(3): 14-16.

[6] KANHABUA N, NORVAG K. Determining time of queries for re-ranking search results[M]. ECDL 2010.

[7] DAKKA W, GRAVANO L, IPEIROTIS P G. Answering general time-sensitive queries[J]. Knowledge & Data Engineering IEEE Transactions on, 2012, 24(2): 220-350.

[8] LIN S, JIN P, ZHAO X, et al. Exploiting temporal information in Web search [J]. Expert Systems with Applications, 2014, 41(2): 331-411.

[9] 张晓娟，陆伟，周红霞. 用户查询中潜在时间意图分析及其检索建模 [J]. 现代图书情报技术，2011(11): 38-43.

[10] JOHO H, JATOWT A, BLANCO R. NTCIR temporalia: a test collection for temporal information access research [M]. Proceedings of the 23rd International Conference on World Wide Web, Seoul, Korea, ACM, 2014.

(责任编辑：杜能钢)

(上接第 11 页)

[2] TURHAN B, MENZIES T, BENER A B, et al. On the relative value of cross-company and within-company data for defect prediction[J]. Empirical Software Engineering, 2009, 14(5):540-578.

[3] RYU D, JANG J, BAIK J. A hybrid instance selection using nearest-neighbor for cross-project defect prediction [J]. Journal of Computer Science and Technology, 2015, 30(5):969-980.

[4] ZIMMERMANN T, NAGAPPAN N, GALL H, et al. Cross-project defect prediction a large scale experiment on data vs. domain vs. process[C]. Joint Meeting of the European Software Engineering Conference and the ACM International Symposium on Foundations of Software Engineering, Amsterdam, Netherlands, 2009: 91-100.

[5] PETERS F, MENZIES T, MARCUS A. Better cross company defect prediction[C]. Working Conference on Mining Software Repositories. San Francisco, USA, 2013:409-418.

[6] 王星，何鹏，陈丹，等. 跨项目缺陷预测中训练数据选择方法[J]. 计算机应用，2016, 36(11):3165-3169.

[7] LU H, CUKIC B, CULP M. Software defect prediction using semi-supervised learning with dimension reduction[C]. Ieee/acm International Conference on Automated Software Engineering. ACM, 2012:314-317.

[8] HERBOLD S. Training data selection for cross-project defect prediction[C]. Proceedings of the 9th International Conference on Predictive Models in Software Engineering. 2013:1-10.

[9] HALL T, BEECHAM S, BOWES D, et al. A systematic literature review on fault prediction performance in software engineering[J]. Software Engineering IEEE Transactions on, 2012, 38(6): 1276-1304.

(责任编辑：陈福时)