现最好,而在其余 3 个项目 ArgoUML、Maven 和 Kylin 上的表现不相上下.经过对实验数据的分析,我们发现,修复一个缺陷,在 Ant 项目中平均要修改 4 个方法体;在 Kylin 项目中平均要修改 8 个方法体;Maven 项目中平均要修改 5.4 个方法体;Argouml 项目中平均要修改 6.2 个方法体.就修复一个缺陷所要修改方法体的数量方面,明显 Ant 项目需要修改的数量最少,这在一定程度上降低了缺陷定位的难度,使得 MethodLocator 在 Ant 项目上具有较好的表现.

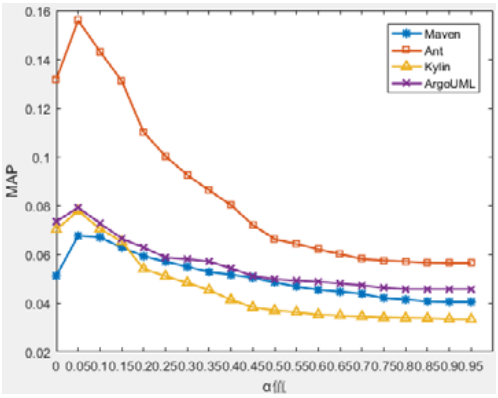


Fig.6 Effect of expansion coefficient α on MAP value in MethodLocator

图 6 MethodLocator 中,扩充系数α对 MAP 值的影响

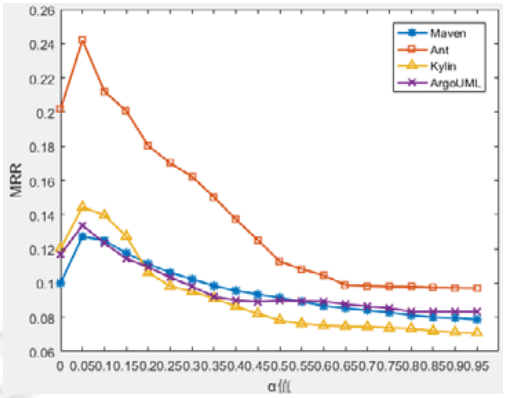


Fig.7 Effect of expansion coefficient α on MRR value in MethodLocator

图 7 MethodLocator 中,扩充系数α对 MRR 值的影响

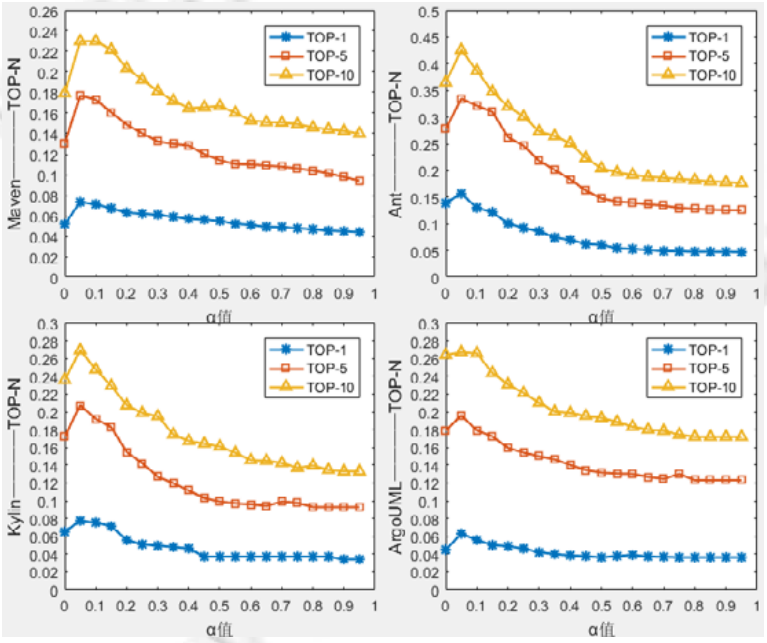


Fig.8 Effect of expansion coefficient α on ToP-N value in MethodLocator

图 8 MethodLocator 中,扩充系数α对 ToP-N 值的影响

4.2 对研究问题2的分析

本文将传统的文件级别的缺陷定位方法 BugLocator 应用到方法级别进行实验,以考察文件级别定位方法

被运用到方法级别定位时的表现.当α=0.05 时,与本文提出的方法 MethodLocator 相比,二者的不同之处主要在于,BugLocator 对方法体进行向量表示时直接采用 tfidf 的向量表示方法,且没有对方法体进行扩充;MethodLocator 在对方法体进行向量表示时,采用将 word2vec 和 tfidf 相结合的方法,并且对方法体进行扩充.表 4 显示了 MethodLocator 和 BugLocator 在本文选择的 4 个项目上的实验结果.

Table 4 MethodLocator and BugLocator comparison of experimental results

表 4 MethodLocator 和 BugLocator 实验结果对比

项目名称	方法名称	ToP-1	ToP-5	ToP-10	MAP	MRR
Ant	BugLocator	0.139	0.278	0.365	0.134	0.210
	MethodLocator, α=0.05	**0.157**	**0.335**	**0.426**	**0.156**	**0.242**
	MethodLocator, α=0	**0.139**	**0.278**	**0.366**	0.132	0.202
Maven	BugLocator	0.053	0.138	0.185	0.054	0.101
	MethodLocator, α=0.05	**0.073**	**0.177**	**0.230**	**0.068**	**0.127**
	MethodLocator, α=0	0.052	0.130	0.180	0.051	0.100
Kylin	BugLocator	0.068	0.176	0.241	0.071	0.128
	MethodLocator, α=0.05	**0.077**	**0.207**	**0.269**	**0.078**	**0.144**
	MethodLocator, α=0	0.065	0.172	0.236	0.070	0.120
ArgoUML	BugLocator	0.046	0.183	**0.273**	0.073	0.118
	MethodLocator, α=0.05	**0.063**	**0.196**	0.267	**0.079**	**0.134**
	MethodLocator, α=0	0.045	0.179	0.264	**0.073**	0.117

从表中可以看到,MethodLocator 具有比 BugLocator 更好的效果.具体来说,在 MAP 指标上,MethodLocator 较之 BugLocator 在 4 个项目上分别提高了 16.4%、25.9%、9.9%、8.2%;在 MRR 指标上,MethodLocator 较之 BugLocator 在 4 个项目上分别提高了 15.2%、25.7%、12.5%、13.6%;在 TOP-*N* 指标上,以 ToP-1 为例,MethodLocator 较之 BugLocator 在 4 个项目上分别提高了 12.9%、37.7%、13.2%、37.0%.方法体相对于源代码文件来说,文本内容及可包含信息明显较少,对于传统文件级方法运用到方法级别定位时,直接对方法体进行向量表示并进行定位,就会因为方法体信息表示不足而取得较差的结果.

当α=0 时,表 4 中展示的即为不对方法体进行扩充时的缺陷定位效果.可以发现,在 20 个指标中,有 3 个是与 BugLocator 相同的,只有在 1 个指标上比 BugLocator 表现好.所以,只使用新的向量表示方法而不对方法体进行扩充,MethodLocator 缺陷定位效果并没有优于基准方法,进一步显示出方法体扩充的重要性.关于采用新的向量表示方法与方法体扩充对准确率的影响更为详细的测算问题,则需要设置更为精确的对比实验来探讨.

4.3 对研究问题3的分析

BLIA 1.5 方法是一种综合了多方面信息源进行方法级缺陷定位的方法.从文献[25]中各信息源的影响分析中可以看到,如果一个缺陷报告包含有缺陷的堆栈信息,那么在缺陷定位中,堆栈信息的分析将起到主要作用,其次才是源代码与缺陷报告之间的相似性;如果一个缺陷报告中不包含堆栈信息,在缺陷定位时,源代码与缺陷报告之间的相似性则起主要作用.与本文所提出的 MethodLocator 相比,BLIA 1.5 考虑的信息源较多.但是由于多源信息之间往往会存在相互冲突和噪声情况,因此,BLIA 1.5 在方法级别缺陷定位上的性能需要进一步验证.

从表 5 可以看出,MethodLocator 方法相对于 BLIA 1.5 方法在方法级别的软件缺陷定位方面达到了较好的效果.具体来说,在 MAP 指标上,MethodLocator 较之 BLIA 1.5 在 4 个项目上分别提高了 16.4%、23.6%、6.9%、3.9%;在 MRR 指标上,MethodLocator 较之 BLIA 1.5 在 4 个项目上分别提高了 14.7%、24.5%、10.8%、11.7%;在 TOP-*N* 指标上,以 ToP-1 为例,MethodLocator 较之 BLIA 1.5 在 4 个项目上分别提高了 12.1%、32.7%、8.5%、28.6%.尽管 BLIA 1.5 方法单独考虑了缺陷报告中的堆栈信息,将其与总结、描述和评论文本中的自然语言文本分开处理,但是首先,在本文所考察的项目中,所有缺陷报告中包含堆栈信息的缺陷报告所占比例并不大,普遍为 5%~10%之间,这些堆栈信息显然并不能在缺陷定位的结果中起到决定性的作用;其次,即使缺陷报告中含有堆栈信息,然而,由于堆栈信息所包含信息往往从最表面调用的函数代码追踪到最深层出现问题的源代码,整个堆栈轨迹的路径较长,从而导致对可能发生缺陷的源代码的误判.例如,在本文考察的 ArgoUML 项目缺陷报告 4 173 中,它包含的堆栈信息由表及里的追踪到了 11 个类的 14 个函数.也就是说,11 个类和 14 个函数以及它们

依赖的类和函数都可能是造成该缺陷的原因.进一步来说,通过本文调研发现,堆栈信息一般在文件级别的粗粒度缺陷定位中有较好的效果[14,16].但是对于方法级别的细粒度定位,由于堆栈信息在引起缺陷的方法的指向性信息上弱化,因此并未见利用其改善缺陷预测或者定位的研究成果.由于以上诸多原因,在本文的实验中,堆栈信息并不能显著提高方法级别的缺陷定位的效果.

Table 5 MethodLocator and BLIA 1.5 comparison of experimental results

表 5 MethodLocator 和 BLIA 1.5 实验结果对比

项目名称	方法名称	ToP-1	ToP-5	ToP-10	MAP	MRR
Ant, α=0.05	BLIA 1.5	0.140	0.308	0.405	0.134	0.211
	MethodLocator	**0.157**	**0.335**	**0.426**	**0.156**	**0.242**
Maven, α=0.05	BLIA 1.5	0.055	0.136	0.183	0.055	0.102
	MethodLocator	**0.073**	**0.177**	**0.230**	**0.068**	**0.127**
Kylin, α=0.05	BLIA 1.5	0.071	0.184	0.260	0.072	0.130
	MethodLocator	**0.077**	**0.207**	**0.269**	**0.078**	**0.144**
ArgoUML, α=0.05	BLIA 1.5	0.049	0.185	**0.274**	0.076	0.120
	MethodLocator	**0.063**	**0.196**	0.267	**0.079**	**0.134**

5 结 论

对于任何软件项目,软件缺陷的出现都是不可避免的.为了提高软件质量和用户满意度,必须快速而及时、准确而有效地修复软件在实际使用中的缺陷.面对一个缺陷报告,软件开发人员需要找到缺陷发生的位置并修复该缺陷,这是一项费时、费力的任务.为了帮助软件开发人员能快速找到缺陷发生的位置,本文提出了一种基于信息检索的方法级别的缺陷定位方法 MethodLocaor.不同于传统的文件级别的定位方法,该方法旨在缩小缺陷定位的粒度而尽量减少缺陷修复人员的修复工作量.通过词向量方法和 TF-IDF 方法,MethodLocaor 实现了源代码方法的初步向量表示.通过方法体扩充方法,MethodLocaor 解决了源代码方法在文本表示上的稀疏性.在 4 个实际项目中的实验表明,MethodLocaor 的定位效果受到方法体扩充系数α的影响,在本文实验的数据集上,α在 0.05 处定位效果最优;与传统文件级定位方法 BugLocator 应用到方法级别的效果相比,MethodLocaor 具有更好的定位性能;与方法级定位方法 BLIA 1.5 方法相比,MethodLocaor 有更好的表现.

在未来的工作中,我们将综合考虑缺陷报告和源代码方法的丰富信息来进一步提高 MethodLacator 在方法级别上的缺陷定位效果.具体来说,在缺陷报告向量表示方面,我们将考虑缺陷报告提交人员的级别、缺陷报告提交时间、超文本链接信息、堆栈信息和代码信息等;在源代码方法方面,我们将考虑方法的圈复杂度、方法之间的依赖关系、开发人员版本信息等.目的是研究有关缺陷报告和源代码方法的全方位信息,并利用其来进行软件方法级别的缺陷定位,以期得到效果更好的缺陷定位方法.

References:

[1] Zhao Y, Leung H, Yang Y, Zhou Y, Xu B. Towards an understanding of change types in bug fixing code. Information & Software Technology, 2017,86:37–53.

[2] Wu W, Zhang W, Yang Y, Wang Q. DREX: Developer recommendation with *K*-nearest-neighbor search and expertise ranking. In: Proc. of the Asia Pacific Software Engineering Conf. Ho CHI Minh, DBLP, 2011. 389–396.

[3] Zhang W, Wang S, Wang Q. BAHA: A novel approach to automatic bug report assignment with topic modeling and heterogeneous network analysis. Chinese Journal of Electronics, 2016,25(6):1011–1018.

[4] Jeong G, Kim S, Zimmermann T. Improving bug triage with bug tossing graphs. In: Proc. of the Joint Meeting of the European Software Engineering Conf. and the ACM Sigsoft Symp. on the Foundations of Software Engineering. ACM Press, 2009. 111–120.

[5] Lukins SK, Kraft NA, Etzkorn LH. Bug localization using latent dirichlet allocation. Information & Software Technology, 2010, 52(9):972–990.

[6] Youm KC, Ahn J, Kim J, Lee E. Bug localization based on code change histories and bug reports. In: Proc. of the Asia-Pacific Software Engineering Conf. 2015. 190–197.

[7] Naish L, Hua JL, Ramamohanarao K. A model for spectra-based software diagnosis. ACM Trans. on Software Engineering & Methodology, 2011,20(3):1−32.

[8] Wang X, Zhang W, Wang Q. Two-phase bug localization method based on defect repair history. Computer Systems & Applications, 2014,23(11):99−104 (in Chinese with English abstract).

[9] Tang M, Zhu L, Zou XC. Document vector representation based on Word2Vec. Computer Science, 2016,43(6):214−217 (in Chinese with English abstract).

[10] Poshyvanyk D, Gueheneuc YG, Marcus A, Antoniol G, Rajlich V. Feature location using probabilistic ranking of methods based on execution scenarios and information retrieval. IEEE Trans. on Software Engineering, 2007,33(6):420−432.

[11] Zhou J, Zhang H, Lo D. Where should the bugs be fixed? More accurate information retrieval-based bug localization based on bug reports. In: Proc. of the ICSE 2012. 2012. 14−24.

[12] Moreno L, Treadway JJ, Marcus A, Shen W. On the use of stack traces to improve text retrieval-based bug localization. In: Proc. of the IEEE Int'l Conf. on Software Maintenance and Evolution. IEEE, 2014. 151−160.

[13] Saha RK, Lease M, Khurshid S, Perry DE. Improving bug localization using structured information retrieval. In: Proc. of the IEEE/ACM Int'l Conf. on Automated Software Engineering. ACM Press, 2015. 345−355.

[14] Saha RK, Lawall J, Khurshid S, Perry DE. On the effectiveness of information retrieval based bug localization for C programs. In: Proc. of the IEEE Int'l Conf. on Software Maintenance and Evolution. IEEE, 2014. 161−170.

[15] Wang S, Lo D. Version history, similar report, and structure: Putting them together for improved bug localization. In: Proc. of the Int'l Conf. on Program Comprehension. ACM Press, 2014. 53−63.

[16] Rahman F, Posnett D, Hindle A, Barr E, Devanbu P. BugCache for inspections: Hit or miss? In: Proc. of the ACM Sigsoft Symp. on the Foundations of Software Engineering (SIGSOFTSoft/FSE 2011). DBLP, 2011. 322−331.

[17] Le TDB, Oentaryo RJ, Lo D. Information retrieval and spectrum based bug localization: Better together. In: Proc. of the Joint Meeting on Foundations of Software Engineering. ACM Press, 2015. 579−590.

[18] Wong CP, Xiong Y, Zhang H, Hao D, Zhang L, Mei H. Boosting bug-report-oriented fault localization with segmentation and stack-trace analysis. In: Proc. of the IEEE Int'l Conf. on Software Maintenance and Evolution. IEEE Computer Society, 2014. 181−190.

[19] Ye X, Bunescu R, Liu C. Learning to rank relevant files for bug reports using domain knowledge. In: Proc. of the ACM Sigsoft Int'l Symp. on Foundations of Software Engineering. ACM Press, 2014. 689−699.

[20] Ye X, Shen H, Ma X, Bunescu R, Liu C. From word embeddings to document similarities for improved information retrieval in software engineering. In: Proc. of the IEEE/ACM Int'l Conf. on Software Engineering. IEEE, 2016. 404−415.

[21] Giger E, D'Ambros M, Pinzger M, Gall HC. Method-level bug prediction. In: Proc. of the ESEM 2012. 2012. 171−180.

[22] Yuan Z, Yu LL, Liu C. Bug prediction method for fine-grained source code changes. Ruan Jian Xue Bao/Journal of Software, 2014, 25(11):2499−2517 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4559.htm [doi: 10.13328/j.cnki.jos.004559]

[23] Hata H, Mizuno O, Kikuno T. Bug prediction based on fine-grained module histories. In: Proc. of the Int'l Conf. on Software Engineering. IEEE, 2012. 200−210.

[24] Wen M, Wu R, Cheung SC. Locus: Locating bugs from software changes. In: Proc. of the IEEE/ACM Int'l Conf. on Automated Software Engineering. IEEE, 2016. 262−273.

[25] Youm KC, Ahn J, Lee E. Improved bug localization based on code change histories and bug reports. Information & Software Technology, 2016,82:177−192.

[26] Phan XH, Nguyen LM, Horiguchi S. Learning to classify short and sparse text & Web with hidden topics from large-scale data collections. In: Proc. of the WWW 2008. 2008. 91−100.

[27] Quan X, Liu G, Lu Z, Ni X, Liu W. Short text similarity based on probabilistic topics. Knowledge and Information Systems, 2010, 25(3):473−491.

[28] Chen M, Jin X, Shen D. Short text classification improved by learning multi-granularity topics. In: Proc. of the Int'l Joint Conf. on Artificial Intelligence (IJCAI 2011). Barcelona, 2011. 1776−1781.

[29] Ma HF, Zeng XT, Li XH, Zhu ZQ. Short text feature extension method of improved frequent term set. Computer Engineering, 2016,42(10):213−218 (in Chinese with English abstract).

[30] Mikolov T, Sutskever I, Chen K, Corrado G, Dean J. Distributed representations of words and phrases and their compositionality. In: Proc. of Int'l Conf. on 27th Conf. on Neural Information Processing Systems (NIPS 2013). Lake Tahoe, 3111−3119.

[31] Joachims T. A probabilistic analysis of the rocchio algorithm with TFIDF for text categorization. In: Proc. of the 14th Int'l Conf. on Machine Learning. Morgan Kaufmann Publishers Inc., 1997. 143−151.

[32] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. In: Proc. of Workshop at 2013 Int'l Conf. on Learning Representation. 2013.

[33] Boureau YL, Bach F, Lecun Y, Ponce J. Learning mid-level features for recognition. In: Proc. of the Computer Vision and Pattern Recognition. IEEE, 2010. 2559−2566.

[34] Zimmermann T, Zeller A. When do changes induce fixes? In: Proc. of the Int'l Workshop on Mining Software Repositories. ACM Press, 2005. 1−5.

附中文参考文献:

[8] 王旭,张文,王青.基于缺陷修复历史的两阶段缺陷定位方法.计算机系统应用,2014,23(11):99−104.

[9] 唐明,朱磊,邹显春.基于 Word2Vec 的一种文档向量表示.计算机科学,2016,43(6):214−217.

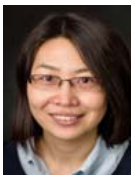
[22] 原子,于莉莉,刘超.面向细粒度源代码变更的缺陷预测方法.软件学报,2014,25(11):2499−2517. http://www.jos.org.cn/1000-9825/4559.htm [doi: 10.13328/j.cnki.jos.004559]

[29] 马慧芳,曾宪桃,李晓红,朱志强.改进的频繁词集短文本特征扩展方法.计算机工程,2016,42(10):213−218.

张文(1981－),男,湖北洪湖人,博士,教授,博士生导师,CCF 专业会员,主要研究领域为软件工程,数据挖掘.

杜宇航(1994－),男,硕士生,主要研究领域为软件工程,数据挖掘.

李自强(1994－),男,硕士生,主要研究领域为软件工程,数据挖掘.



杨叶(1977－),女,博士,教授,博士生导师,主要研究领域为软件工程.