智能软件系统数据驱动的软件测试方法与技术

缺陷报告分派、安全缺陷报告识别及故障定位

（测试的后期维护）

（一）立项依据与研究内容：

1．项目的立项依据（研究意义、国内外研究现状及发展动态分析，需结合科学研究发展趋势来论述科学意义；或结合国民经济和社会发展中迫切需要解决的关键科技问题来论述其应用前景。附主要参考文献目录）；

1.1 研究意义

智能软件系统的特点是不确定性和概率性。对大数据具有依赖性、随机性的输入/输出、难以预测所有应用场景、需要从过去的行为中不断自我学习。采用的机器学习核心算法可能在种类和复杂性上有所不同，并且随着软件系统的应用可能组合更多的算法，具有动态变化的软件需求。通常被应用于其他更大的应用程序以满足最终的业务目标，因此需要第三方系统的接口。这些因素导致了智能软件系统开发与测试的复杂性，传统的软件开发和测试方法，将不能满足这些系统的需求。

当前，以互联网和移动通信为纽带，人类群体、大数据、物联网已经实现了广泛和深度的互联，使得人类群体智能日益发挥越来越重要的作用。越来越多的来自世界各地的开发人员都在参与开发同一软件项目。智能系统开发方法从封闭和计划走向开放和竞争。

这些项目通常具有动态变化的软件需求，团队组织松散以及开发人员自由协作等特点。在协作过程中，许多与软件开发相关的信息，包括电子邮件系统，配置管理工具，缺陷跟踪系统等，都可以记录在软件仓库中。如果能够利用机器学习技术来针对这些数据进行分析，则有助于更好地理解测试行为，改进测试自动化。

1.1.1 基于开发者社交网络的缺陷报告分派

在传统的软件开发中，开发人员之间的协作和沟通一般是基于面对面交流来解决问题。而现如今的软件开发流程中会呈现出更多新的特征。首先，随着软件的大小和复杂性的增加，软件团队中的开发人员数量也在增长。其次，开发人员的协作和沟通主要是基于互联网而不是面对面的谈话。第三，不同的开发人员会参与软件的不同开发阶段。因此，挖掘开发人员和开发人员之间的潜在合作关系

能够更有利于我们分析不同开发者所擅长的知识领域，然后通过不同的知识领域对所有开发者进行分类。因此，利用开发人员的合作信息来改进工程任务是当今软件领域的一项新的挑战。

在发现软件项目中的异常行为后，测试人员或用户将会提交缺陷报告。特别是随着当前软件系统的大小和复杂性的增加，测试人员和用户将越来越多的缺陷提交给项目的缺陷跟踪系统。例如，在接近发布日期时，Eclipse 缺陷存储库每天都会接收到大约 200 个缺陷报告提交，对于 Debian 项目，这个数字大约是 150 个。

这些缺陷报告必须进行分类并分配给具有相关经验的开发人员来处理。如果采用人工的方法将缺陷报告分派将会耗费大量时间和人力，并且可能将缺陷报告被分配给不相关的开发者，导致这些缺陷的修复延期。但是假如能够充分利用缺陷追踪系统上的信息：已修复的历史记录、缺陷报告的分类信息和文本信息等，挖掘开发人员之间的潜在合作关系,这样在新的缺陷报告出现后，我们能够迅速找到相关知识领域的合适的开发人员进行修复。从而可以将大量人力投入到软件本身的问题上，从而提升软件软件质量。

1.1.2 安全漏洞相关的缺陷报告识别

软件安全漏洞是在软件设计与实现过程中存在的一些容易被恶意攻击者所利用、对软件安全构成威胁的缺陷或不足。常见的软件安全漏洞包括缓冲区溢出、内存泄漏、整数溢出等。随着软件规模的不断增大，以及软件复杂度的不断提高，健壮的软件系统已经很难被设计和实现，所以软件漏洞的存在是不可避免的。因此，如何快速而准确地发现软件安全漏洞一直以来是信息安全领域的研究热点。

由此可见，软件安全漏洞检测对于保证软件系统安全具有十分重要的意义。而代码和缺陷报告作为软件开发和维护的重要组成部分，能够为软件安全漏洞检测提供重要的信息来源。尽管缺陷报告在被提交时需要被标明该缺陷是否是安全漏洞，但是由于对信息安全知识的欠缺，报告提交者往往无法正确的区分安全漏洞与常规的软件缺陷，因此许多安全漏洞常常被错误地标记为与安全无关的缺陷，从而导致这些安全漏洞无法被及时地修复，容易被恶意攻击者所利用。因此，从海量的缺陷报告中找出与安全漏洞相关的缺陷报告（以下简称“安全缺陷报告”）具有十分重要的实际意义。

1.1.2 基于缺陷报告的故障定位

当开发者接收到具体的缺陷报告后，需要去查找相应的缺陷源代码，一般情况下通常需要重现异常行为并执行代码审查才能找到原因。

但是，缺陷报告的多样性和参差不齐的报告质量可能会使这个过程变得很困难。缺陷报告中经常缺少基本信息，例如缺陷报告中的自然语言陈述与软件系统

中的技术术语之间的词汇不匹配问题，这会在一定程度上限制了基于简单词汇匹配分数的排名方法的准确性。

为了定位缺陷，开发人员不仅需要使用他们的领域知识分析缺陷报告，还需要从同行开发人员和用户收集信息。尤其当文件和报告的数量很大时，比如说由数百甚至数千个文件组成的大型项目，使用人工查找和分析缺陷的产生原因可能非常耗时。

因此，如果能通过解析缺陷报告自动定位到缺陷源代码，则有助于提高软件维护的效率。

1.2 国内外研究现状及发展动态分析

1.2.1 缺陷报告自动分派方法的研究现状

目前国内外关于缺陷报告自动分派的研究主要基于以下几类：

（1）基于机器学习的缺陷报告分派。Cubranic D 等人[1]第一次使用机器学习（朴素贝叶斯分类器）的方法自动将缺陷报告分配给合适的开发者。S.N.等人[2]将缺陷报告的文本字段转换为基于 TF-IDF 和潜在语义索引的多维“术语-文档”矩阵，然后利用不同的机器学习方法进行缺陷报告分类。X.Xia 等人[3]将缺陷分类作为一个多标签分类任务，其中缺陷报告是数据实例，每个开发人员对应一个类标签。他们根据每个新缺陷报告的 k-近邻来预测合适的缺陷修复人员，并通过分析开发人员的专业知识增强了缺陷分类的性能。

（2）基于深度学习的缺陷报告分派。Lee 等人[4]和 S.Mani 等人[5]将缺陷报告的文本字段作为深度学习模型的输入，输出将缺陷报告分配给每个开发者的概率。不同之处在于前者使用了 CNN（Convolutional Neural Network）网络提取缺陷报告的文本语义特征，而后者使用了双向 LSTM（Long Short-Term Memory）机制。

（3）基于信息检索的缺陷报告分派。X.Xie 等人[6]提出了一种名为 DRETOM（即 Developer Recommendation based on Topic Models）的基于主题的匹配方法，该方法基于历史缺陷报告修复记录构建主题模型，模拟了开发人员在缺陷报告修复活动中的兴趣和专业知识。当来了新的缺陷报告后，DRETOM 根据这些开发人员的兴趣和专业知识，推荐了一份可能会解决新缺陷的开发人员排名列表。T.Zhang 等人[7]建立了一个经验模型和一个概率模型，然后通过这两个模型组合成混合缺陷分类算法，用以推荐合适的开发人员来修复缺陷。

1.2.2 安全缺陷报告识别

关于安全缺陷报告识别，近些年国外开始有一些研究，而国内尚无相关的研究。目前关于安全缺陷报告识别主要集中在以下两个问题的研究：

（1）文本挖掘方面的研究

安全缺陷报告识别主要是利用缺陷报告中的自然语言信息，如何有效地挖掘缺陷报告中的文本信息是一个关键的研究问题。Gegick[8]根据缺陷报告中的自然语言描述信息，利用词袋模型将所有缺陷报告表示成一个词项-文档矩阵，并将该矩阵和相应的标签作为输入来训练统计模型，从而进行安全缺陷报告识别。Behl[9]同样利用词袋模型表示缺陷报告，并且引入了 TF-IDF 值作为词的权重并通过朴素贝叶斯模型来进行安全缺陷报告识别。Chawla[10]利用了缺陷报告所包含的语义信息，通过 TF-IDF、LSI 模型，以及多项式朴素贝叶斯模型对缺陷报告进行分类。Zou[11]充分利用缺陷报告的文本信息和非文本信息训练模型，从而进行安全缺陷报告识别。

(2)类别不平衡方面的研究

由于在安全缺陷报告识别的任务中，训练数据存在严重的类别不平衡的问题，即安全缺陷报告的数量要远远小于非安全缺陷报告的数量，这会使得预测模型无法实现预期的效果，因此一些研究在类别不平衡处理方面进行了探索。Yang[12]利用词袋模型将缺陷报告表示成向量，并分别利用四种不平衡数据的处理方法（随机欠采样、随机过采样、合成少数类样本的过采样、代价矩阵调节方法）对训练集进行处理，再利用经过处理的训练集训练分类器来识别具有较大影响的缺陷报告。Zhou[13]利用 NLP 和机器学习技术来识别缺陷报告和提交信息中的安全问题，他们提出了 K 折叠加算法来集成多个个体分类器，从而缓解数据集中类别不平衡的影响。Postojanova[14]提取了三种缺陷报告的特征向量：二元词袋频率（BF）、词频（TF）和 TF-IDF，并分别利用监督学习和基于异常监测的非监督学习算法来识别安全缺陷报告，非监督学习克服了监督学习需要人工标注数据以及可能出现类别不平衡问题的缺点。

Peters[15]发现一些非安全缺陷报告同样包含和安全相关的关键词，这些非安全缺陷报告相当于对模型的训练引入了噪音，势必会影响模型的训练效果，增大了模型将安全缺陷报告误标记为非安全缺陷报告的概率，而这种不利的影响在类别不平衡的情况下又被扩大。因此他们提出了一个框架 FARSEC 用来进行安全缺陷报告识别。该框架在训练预测模型前将含有安全关键词的非安全缺陷报告从训练集中移除，从而提高模型对安全缺陷报告的识别效果。该方法从安全缺陷报告中提取 TF-IDF 值最高 100 个词作为安全相关的关键词，并利用这 100 个安全关键词来过滤非安全缺陷报告，同时还利用这些安全关键词将每一个缺陷报告表示成一个 100 维的特征向量作为模型的输入。然而，这种方法存在一定的问题。首先，这些 TF-IDF 值较高的词未必就是和安全相关的词，如果这些安全关键词不准确，

那么对非安全缺陷报告的过滤效果就会大打折扣。其次，用提取的 100 个安全关键词来表示缺陷报告，会造成缺陷报告的向量表示具有很大的稀疏性，这是因为在一份缺陷报告中很可能只出现少数几个安全相关的关键词。

1.2.3 基于缺陷报告的故障定位的研究现状

（1）基于信息检索技术的故障定位。Gay 等人，采用信息检索技术，根据缺陷报告自动搜索相关文件[16]。他们将初始缺陷报告视为查询，并根据与查询的相关性对源代码文件进行排名。

Ye X 等人[17]结合了 4 种相关计算方法对可能存在缺陷的源代码文件进行排名，这四种计算方法包括 1）通过简单的空间向量模型来计算源代码和缺陷报告的相似性。2）利用 API 描述来弥补缺陷报告和源代码之间的词汇差距。3）利用先前已解决的缺陷报告的创建时间作为辅助计算。4）采用协同过滤的方法，通过检查源代码文件之前所涉及的所有缺陷报告并将其提取摘要，然后和新的缺陷报告做余弦相似度计算。

2016 年 Ye X 等人[18]又对之前的方法提出了改进，扩充了另外两个对源代码排名有参考价值的方法：1）将源代码文件分解为类库，方法库，变量库，评论库，然后分别与新缺陷报告的摘要和描述做相似度计算。2）通过源代码文件的依赖关系建立文件网络图，然后用类似 PageRank 的方法计算新缺陷报告与源文件的相似度。

（2）基于深度学习的故障定位。An Ngoc Lam 等人[19]使用深度学习进行故障定位。他们使用改进过的向量空间模型（VSM，Vector Space Model）计算缺陷报告和源文件之间的相似度，并用深度神经网络（DNN，Deep Neural Networks）来学习缺陷报告的具体术语和源文件文本标记之间的关联，最后将两者相结合，给出源代码文件和缺陷报告的相似度排名。

1.2.3 难点科学问题

自动化测试有大量的历史数据，如果能够利用机器学习来针对这些数据进行分析，则有助于更好地理解测试行为，改进测试自动化。已有研究为本项目研究缺陷指派和基于缺陷报告的故障定位方法奠定了良好的基础，但仍存在以下问题需要进一步研究与解决。

⑴ 如何挖掘开发人员之间的潜在合作关系，这样在新的缺陷报告出现后，能够迅速找到相关知识领域的合适的开发人员进行修复。

尽管之前很多人提出了各种方法来使错误分类过程自动化，但这些方法大部分只能预测一名开发人员来修复新近的缺陷报告。而修正缺陷报告是一项协同的

工作，即许多开发者贡献了他们修复缺陷报告的经验。因此，如何找出这些具有相关领域缺陷修复经验的人至关重要。

（2）如何有效识别安全缺陷报告?

已有的方法仅根据安全关键词来过滤的方法识别安全缺陷报告,缺少对缺陷报告中语义信息的考虑,因此需要研究以缺陷报告间的相似度为依据的在非安全缺陷报告的过滤方法和缺陷报告的语义信息表示。

（3）如何结合领域知识和缺陷修复的历史信息，更加准确地定位软件故障，辅助软件调试、并优化回归测试范围？

缺陷报告和源代码之间的词汇存在差异，需要进一步研究从源代码中提取领域知识，准确计算缺陷报告和源代码之间的相似度。

已修复的缺陷报告和其修复的源代码文件建立了关联关系，可以为修复相似缺陷提供启发信息。在故障定位时也应充分考虑该信息。

本项目综合运用程序分析、智能软件、信息检索等技术对以上难点问题进行研究和解决。项目组成员近年来一直从事该领域相关理论和技术的研究。本项目的研究成果可应用于工业软件的自动化测试和调试领域，因此具有重要的理论研究意义和应用价值。

参考文献：

[1] Cubranic D.Automatic bug triage using text categorization[C]// Proc. International Conference on Software Engineering & Knowledge Engineering. 2004:92-97.

[2] S. N. Ahsan, J. Ferzund, and F. Wotawa, “Automatic software bug triage system (BTS) based on latent semantic indexing and support vector machine,” 4th International Conference on Software Engineering Advances, ICSEA 2009, Includes SEDES 2009: Simposio para Estudantes de Doutoramento em Engenharia de Software, pp. 216–221, 2009.

[3] X. Xia, D. Lo, X. Wang, and B. Zhou, “Accurate developer recommendation for bug resolution,” Proceedings - Working Conference on Reverse Engineering, WCRE, pp. 72–81, 2013.

[4] S.-r. Lee, M.-j. Heo, C.-g. Lee, and M. Kim, “Applying Deep Learning Based Automatic Bug Triager to Industrial Projects,” pp. 926–931, 2017.

[5]S. Mani, A. Sankaran, and R. Aralikatte, “DeepTriage: Exploring the Effectiveness of Deep Learning for Bug Triaging,” 2018. [Online].

[6]X. Xie, W. Zhang, Y. Yang, and Q. Wang, “DRETOM: Developer Recommendation based on Topic Models for Bug Resolution,”Proceedings of the 8th International Conference on Predictive Models in Software Engineering (PROMISE), pp. 19–28, 2012.

[7] T. Zhang and B. Lee, “A hybrid bug triage algorithm for developer recommendation,” Proceedings of the 28th Annual ACM Symposium on Applied Computing - SAC ’13, p. 1088, 2013.

[8] Gegick M, Rotella P, Xie T. Identifying security bug reports via text mining: An industrial case study[C]//Mining software repositories (MSR), 2010 7th IEEE working

conference on. IEEE, 2010: 11-20.
[9] Behl D, Handa S, Arora A. A bug mining tool to identify and analyze security bugs using naive bayes and tf-idf[C]//Optimization, Reliabilty, and Information Technology (ICROIT), 2014 International Conference on. IEEE, 2014: 294-299.
[10] Chawla I, Singh S K. Automatic bug labeling using semantic information from LSI[C]//Contemporary Computing (IC3), 2014 Seventh International Conference on. IEEE, 2014: 376-381.
[11] Zou D, Deng Z, Li Z, et al. Automatically Identifying Security Bug Reports via Multitype Features Analysis[C]//Australasian Conference on Information Security and Privacy. Springer, Cham, 2018: 619-633.
[12] Yang X, Lo D, Huang Q, et al. Automated identification of high impact bug reports leveraging imbalanced learning strategies[C]//Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual. IEEE, 2016, 1: 227-232.
[13] Zhou Y, Sharma A. Automated identification of security issues from commit messages and bug reports[C]//Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. ACM, 2017: 914-919.
[14] Goseva-Popstojanova K, Tyo J. Identification of Security Related Bug Reports via Text Mining Using Supervised and Unsupervised Classification[C]//2018 IEEE International Conference on Software Quality, Reliability and Security (QRS). IEEE, 2018: 344-355.
[15] Peters F, Tun T, Yu Y, et al. Text Filtering and Ranking for Security Bug Report Prediction[J]. IEEE Transactions on Software Engineering, 2017.
[16] G. Gay, S. Haiduc, A. Marcus and T. Menzies, On the use of relevance feedback in IR-based concept location, Proc. the 25th IEEE International Conference on Software Maintenance, Edmonton, Alberta, Canada, p.351-360, September 2009.
[17] Ye X, Bunescu R, Liu C. Learning to rank relevant files for bug reports using domain knowledge[C]// ACM Sigsoft International Symposium on Foundations of Software Engineering. ACM, 2014:689-699.
[18] Ye X, Bunescu R, Liu C. Mapping Bug Reports to Relevant Files: A Ranking Model, a Fine-Grained Benchmark, and Feature Evaluation[M]. IEEE Press, 2016.
[19] An N L, Nguyen A T, Nguyen H A, et al. Combining Deep Learning with Information Retrieval to Localize Buggy Files for Bug Reports (N)[C]// Ieee/acm International Conference on Automated Software Engineering. IEEE, 2016:476-481.
[20] Blondel V D, Guillaume J L, Lambiotte R, et al. Fast unfolding of communities in large networks[J]. Journal of Statistical Mechanics, 2008, 2008(10):155-168.
[21] Sun C, Lo D, Khoo S C, et al. Towards more accurate retrieval of duplicate bug reports[C]//Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering. IEEE Computer Society, 2011: 253-262.

2．项目的研究内容、研究目标，以及拟解决的关键科学问题（此部分为重点阐述内容）；

2.1 研究内容

（1） 基于社交网络的缺陷报告自动分派方法

1）数据提取

从数据仓库中提取有关开发者的信息，筛选出主要的列，其中包括缺陷描述，缺陷摘要，缺陷修复者，缺陷评论者等。

2）开发者社交网络构建

根据开发者对于同一缺陷的评论关系建立一个完整的开发者社交网络，以便进行社区检测。

3）开发者社区检测

当建立完开发者社交网络后，我们希望将具有相同开发领域经验的人分配到一个社区内，以便在缺陷报告分配时，可以指定给对应的社区来处理缺陷。所以这里使用一定的算法将开发者社交网络进行社区划分。

4）开发者能力评估

根据开发者以往修复的缺陷数目以及该开发者在网络节点中的影响力建立一个关于开发者能力的评价指标，并将开发者按能力评价指标降序排列。

5）缺陷报告分派

根据 1）中所提取的数据，建立一个缺陷分派的预测模型，新缺陷报告通过该预测模型的计算后能够分派给合适的社区，然后再从该社区找到排名最高的开发者来进行修复。

（2） 安全缺陷报告识别方法

在训练模型前将与安全缺陷报告相似度较高的非安全缺陷报告过滤掉，再将过滤后的非安全缺陷报告和所有的安全缺陷报告作为训练集并转化为向量表示，从而训练模型并预测新的缺陷报告是否为安全缺陷报告。

具体研究内容如下：

1）非安全缺陷报告的过滤方式。之前的研究将安全相关的关键词作为过

滤的依据，然而由于难以找到合适的安全关键词，所以无法取得较好的效果。因此如何通过合理地过滤非安全缺陷报告来去除噪音并解决类别不平衡问题是本课题的关键研究问题。

2）缺陷报告的向量表示。之前的研究利用词袋模型，使用与安全相关的关键词作为特征集来提取缺陷报告的向量表示。然而，这种表示方式不仅会造成数据稀疏的问题，也无法准确地表达缺陷报告包含的语义信息。因此探索更合适的文本表示方法也是本课题的重要研究内容。

3）分类器的选取。由于本课题数据集的安全缺陷报告的数量较少，经过对非安全缺陷报告的欠采样后，训练集的样本数也较小，因此无法训练有效的深度学习模型，适用于安全缺陷报告识别任务的分类器有待进一步探索。

（3） 基于缺陷报告的故障定位方法

开发者收到具体的缺陷报告后，接下来就要从源代码文件中查找与该缺陷报告相似度最高的源代码文件。

1）源代码文件与缺陷报告的相似度计算

解析源代码，在抽象语法树基础上，提取类名、方法名、变量名以及评论等并形成领域语料库，然后对缺陷报告提取词向量，求解相应的 TF-IDF 值，然后使用余弦相似度一一计算所有源文件代码和新缺陷报告的相似值。

2）相似缺陷报告的修复历史分析

因为相似的缺陷报告可能会涉及到修复同一源文件，所以我们结合以往的已修复的相似缺陷报告、新缺陷报告以及所有的源代码文件，加以分析，计算所有源代码文件与新缺陷报告的相似值。

3）定位可疑缺陷代码文件

结合 1）2）所述的排名计算值，我们对这两个值分别赋予一定的权重并相加，得到最终的可疑缺陷文件排名。

2.2 研究目标

智能系统开发的测试和维护过程中产生大量的历史数据，本项目研究如何充分挖掘可用的信息，提高自动化测试和维护的效率，帮助软件开发团队提高生产力，提升软件质量。

（1）缺陷报告分派的预期目标：建立基于社交网络的缺陷报告自动分派模型，挖掘开发人员之间的潜在合作关系，为缺陷报告，指派相关知识领域的合适的开发人员进行修复。

（2）安全缺陷报告识别的目标:能够在大量缺陷报告中有效识别安全相关的缺陷报告。

（3）故障定位的预期目标：建立基于缺陷报告的故障定位模型，辅助开发人员快速识别和修复缺陷。

2.3 拟解决的关键科学问题

（1）如何挖掘开发人员之间的潜在合作关系，并用缺陷修复的历史数据训练模型，对于新的缺陷报告，推荐相关知识领域的合适的开发人员进行修复。

（2） 如何考虑缺陷报告的语义信息，准确识别安全相关的缺陷报告？

（3）如何结合领域知识和缺陷修复的历史信息，更加准确地定位软件故障，辅助

软件调试、并优化回归测试范围？

3．拟采取的研究方案及可行性分析（包括研究方法、技术路线、实验手段、关键技术等说明）；

3.1 研究方法和总体技术路线

基于开发者社交网络的缺陷报告分派及缺陷定位总体技术路线如图 1 所示。

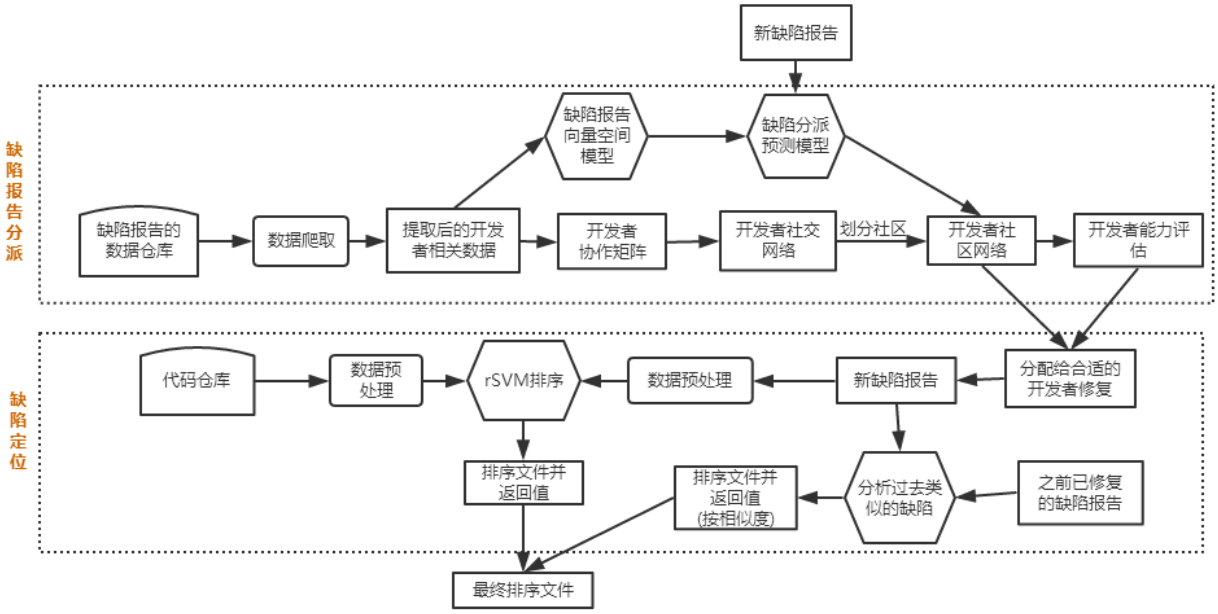

图 1 基于开发者社交网络的缺陷报告分派及故障定位总体技术路线

3.1.1 基于开发者社交网络的缺陷报告分派的研究方案

主要研究技术路线如下：首先 1）从数据仓库中提取开发者的相关信息，根据其中开发者以往的合作关系来构建开发者社交网络，然后 2）应用社区检测算法来划分开发者社区。接着 3）根据社区开发者的所有修复的缺陷数以及该开发者在网络中的中介值(betweenness)为每个社区的开发人员排名。4）使用机器学习并借助缺陷报告的向量空间模型建立一个预测模型，用以预测缺陷报告会被分派到哪个开发者社区，当收到新的缺陷报告后，将其通过预测模型分配到对应的社区，然后再根据开发者能力选择最佳修复者。

以下阐述缺陷报告分派的具体研究方案：

（1）提取数据，构建向量空间模型

使用缺陷报告的摘要作为文本表示。第一步是构建一个缺陷报告的向量空间表示。假设有一组缺陷报告，$B = b_1, ..., b_{|B|}$。每个缺陷报告都有一组术语，$T = t_1, ..., t_{|T|}$。在缺陷报告中为每个术语分配权重，表示缺陷报告中术语出现的频率。过滤掉不必要的词语，包括停用词，标点符号，空格和数字。然后应用卡方（χ^2）特征选择方法来降低向量空间的维数并获得更好的分类结果。对于 χ^2，我们选择 30%

作为语料库中最后一个词的比例，我们选择 χ^2，因为它比文本分类中的其他特征选择方法更好。TF 定义如式（1）所示：

$$词频(TF) = \frac{某个词在文章中的出现次数}{文章的总词数} \quad (1)$$

（2）构建协作矩阵

在构建矢量空间模型之后，提取活动开发人员之间的协作工作，并将结果保存在开发人员-开发人员协作矩阵（DDCM）中。DDCM 矩阵是一个正方形的 d×d 矩阵，其中 d 表示活动开发者的数量。 DDCM [i，j]的值定义如下：

$$DDCM[i, j] = Comments(i, j) + Comments(j, i) + CC(i, j)$$

Comments(i, j)表示开发人员 i 对开发人员 j 的 bug 报告所作的评论数量，Comments（j，i）表示开发人员 j 对开发人员缺陷报告的评论数量，CC（i，j） 代表开发者 i 和 j 都评论过但不属于他们的缺陷报告的数量。主要的依据是，如果两个开发人员在同一个缺陷报告中写了很多评论，并且如果他们对彼此的缺陷报告发表评论的话，那么他们都有解决类似缺陷报告的经验。

（3）创建开发者社交网络

在建立开发者邻接矩阵之后，从 DDCM 矩阵创建一个活跃的开发者网络。每个节点都代表一名活跃的开发人员，如果两个节点（开发人员）之间有协作关系，则其边的权重由 DDCM 两个开发人员之间的协作值决定。

（4）检测开发人员社区

应用 Louvain 算法[20]来从开发者网络中发现开发者社区。Louvain 算法是基于模块度的社区发现算法，该算法在效率和效果上都表现较好，并且能够发现层次性的社区结构，其优化目标是最大化整个社区网络的模块度。

模块度 Modularity 是评估一个社区网络划分好坏的度量方法，它的物理含义是社区内节点的连边数与随机情况下的边数只差。定义如式（2）所示：

$$Q = \frac{1}{2m}\sum_{i,j}[A_{ij} - \frac{k_i k_j}{2m}]\delta(c_i, c_j)$$
$$\delta(u, v) = \begin{cases} 1 & when\ u = v \\ 0 & else \end{cases} \quad (2)$$

其中，A_{ij} 表示节点 i 和节点 j 之间边的权重，网络不是带权图时，所有边的权重可以看做是 1。$k_i=\sum_j A_{ij}$ 表示所有与节点 i 相连的边的权重之和（度数）。C_i 表示节点 i 所属的社区。$m=\frac{1}{2}\sum_{ij} A_{ij}$ 表示所有边的权重之和（边的数目）。公式

（3-2）可进一步化简为公式（3）。

$$Q=\sum_{c}[\frac{\sum in}{2m}-(\frac{\sum tot}{2m})^{2}]$$

（3）

其中 Σin 表示社区 c 内的边的权重之和，Σtot 表示与社区 c 内的节点相连的边的权重之和。

这样模块度也可以理解是社区内部边的权重减去所有与社区节点相连的边的权重和，对无向图更好理解，即社区内部边的度数减去社区所有节点的总度数。

Louvain 具体算法如下：

1）将图中的每个节点看成一个独立的社区，次数社区的数目与节点个数相同；

2）对每个节点 i，依次尝试把节点 i 分配到其每个邻居节点所在的社区，计算分配前与分配后的模块度变化 ΔQ，并记录 ΔQ 最大的那个邻居节点，如果 maxΔQ>0，则把节点 i 分配 ΔQ 最大的那个邻居节点所在的社区，否则保持不变；

3）重复 2），直到所有节点的所属社区不再变化；

4）对图进行压缩，将所有在同一个社区的节点压缩成一个新节点，社区内节点之间的边的权重转化为新节点的环的权重，社区间的边权重转化为新节点间的边权重；

5）重复 1）直到整个图的模块度不再发生变化。

应用该算法后的示例图如图 2 所示。

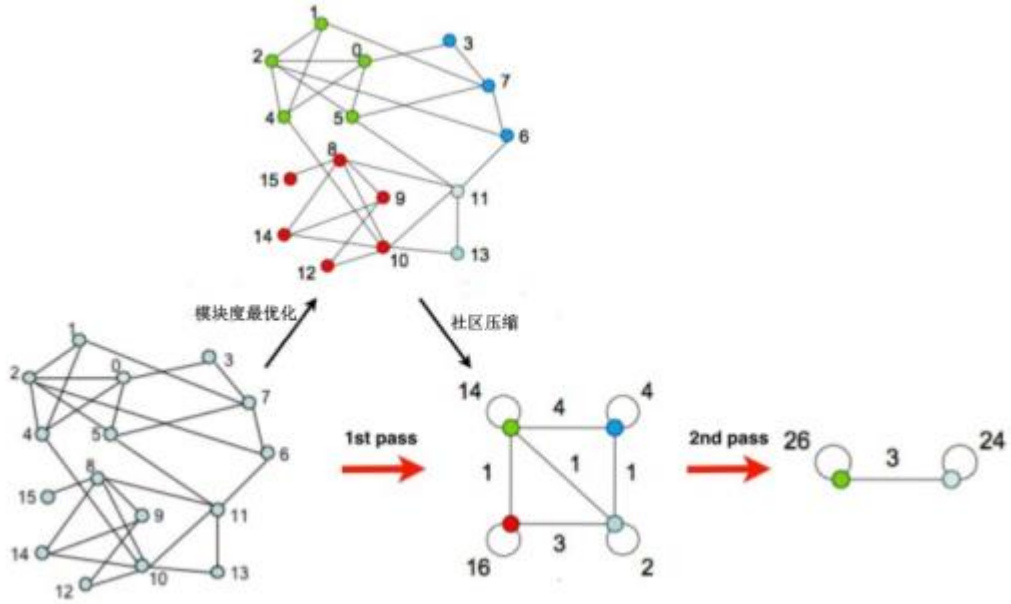

图 2 Louvain 算法流程示例图

（5）评估每个开发者的能力

在分配一个社区来解决一个新的缺陷报告后，我们对该社区的开发者进行能力排名，找到最合适的开发者来修复这个新的错误。开发人员根据他们的经验进行排名。根据以下公式（4）为每个社区中的每个开发人员提供权重：

$$weight(D_i|C_j)=\frac{Fixed(D_i|C_j)}{\sum_{k=1}^{N}Fixed(D_k|C_j)}+\frac{Bet(D_i|C_j)}{\sum_{k=1}^{N}Bet(D_k|C_j)}$$

（4）

其中 Fixed（Di ｜ Cj）表示社区 Cj 中开发者 Di 所修复的类似缺陷报告数量，判断两个缺陷报告是否相似我们使用余弦相似值来计算，所得值为非 0 即为相似报告。Bet（Di｜ Cj）表示社区 Cj 中开发者 Di 的节点介数（网络中所有最短路径中经过该节点的路径的数目占最短路径总数的比例），分母用于得到标准化的权重。该公式的依据是给那些在社区中修正了更多错误，并且是社区中最有影响力的开发者更多的权重。

（6）建立一个预测模型，分派缺陷报告

在检测开发者社区后，我们建立一个预测模型，预测社区解决新的缺陷报告。缺陷分配被制定为一个分类任务，其中实例代表缺陷报告，特征代表报告的独特术语，而类标签代表在解决这个报告时合作的社区。由于上一步我们在开发者网络中发现了密集的社区，我们在类标签中用开发者所在的社区替换相应的开发者标签。

为了构建预测模型，我们可以使用两种广泛使用的机器学习技术，即朴素贝叶斯和随机森林。朴素贝叶斯算法是一种概率分类器，假定所有特征都是独立的，并且使用贝叶斯定理给出一组特征值，从而找到具有最大概率的类。随机森林算法生成许多决策树，使每棵树预测一个类标签，并选择具有多数选票的类标签。本课题暂用随机森林算法。

3.1.2 基于缺陷报告的故障定位的研究方案

主要研究技术路线如下：首先，通过计算源文件和新缺陷报告的相似度，对源代码文件进行排序。步骤包括 1）解析源程序，基于抽象语法树，提取类名、方法名、变量名以及注释等领域知识形成语料库。2）然后对缺陷报告提取词向量，求解相应的 TF-IDF 值。3）利用 rSVM 模型计算源文件和新缺陷报告的相似度，排序各个源代码文件。然后，分析之前已经修复的缺陷报告，将新缺陷报告、已修复的类似缺陷报告以及所有的源代码文件三者组成三层异构图，在此基础上，计算源代码文件的又一排名。最后，结合前面两步中分别得到的排名，赋予两者一定的权重因子，得到最终排名。

具体研究方案如下：

（1）通过计算源文件和新缺陷报告的相似度，对源代码文件进行排序。

将源代码文件和新缺陷报告进行数据预处理后，通过计算每个文件和缺陷报告之间的相似性，根据相似度对文件进行排序并作为返回值。

提出了一种改进的向量空间模型（rVSM）来计算它们之间的相似度。 在经

典的 VSM 中，文档 d 和查询 q 之间的相关性得分是由它们对应的向量表示之间的余弦相似度得到的，定义如式（5）所示。

$$Similarity(q,d)=\cos(q,d)=\frac{\overrightarrow{V_q}\bullet\overrightarrow{V_d}}{\left|\overrightarrow{V_q}\right|\left|\overrightarrow{V_d}\right|}$$

（5）

其中 $\overrightarrow{V_q}$，$\overrightarrow{V_d}$ 分别是文档 d 和查询 q 的术语权重向量。术语权重 w 的计算是基于项频率（tf）和逆文档频率（idf）。基本思想是文档中术语的权重随着其在该特定文档中的出现频率而增加，并且随着其在其他文档中的出现频率而降低。 在经典 VSM 中，tf 和 idf 定义如式（6）所示。

$$tf(t,d)=\frac{f_{td}}{\#terms},idf(t)=\log(\frac{\#docs}{n_t})$$

（6）

其中 f_{td} 是指文档 d 中术语 t 的出现次数，n_t 是指包含术语 t 的文档数，#terms 表示文档 d 中的术语总数，#docs 表示语料库中的文档总数。

$$tf(t,d)=\log(f_{td})+1$$

（7）

在 rVSM 中，我们加以改进，使用等式（7）来定义 tf。 因此，在等式（5）中，文档向量 $\overrightarrow{V_d}$ 中的每个项权重 w 以及其范数 $\left|\overrightarrow{V_d}\right|$ 计算如下式（8）所示：

$$w_{ted}=tf_{td}\times idf_t=(\log f_{td}+1)\times\log(\frac{\#docs}{n_t})$$

$$\left|\overrightarrow{V_d}\right|=\sqrt{\sum_{ted}((\log f_{td}+1)\times\log(\frac{\#docs}{n_t}))^2}$$

（8）

结合上述分析，我们提出了一种新的 rVSM 评分算法如下式（9）：

$$rVSMScore(q,d)=\cos(q,d)$$

$$=\frac{1}{\sqrt{\sum_{teq}((\log f_{tq}+1)\times\log(\frac{\#docs}{n_t}))^2}}$$

$$\times\frac{1}{\sqrt{\sum_{ted}((\log f_{td}+1)\times\log(\frac{\#docs}{n_t}))^2}}$$

$$\times\sum_{teq\cap d}(\log f_{td}+1)\times(\log f_{tq}+1)\times\log(\frac{\#docs}{n_t})^2$$

（9）

（2）基于相似缺陷报告的修复历史分析，对源代码文件进行排序。

先构建一个如图 3 所示的三层异构网络图。

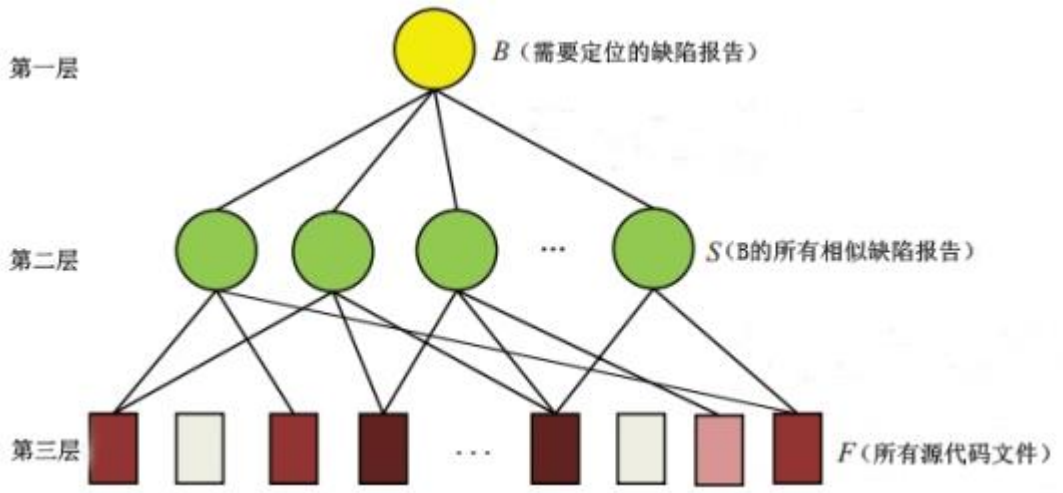

图 3　三层异构网络图

其中，第一层包含一个结点 B，代表新来的缺陷报告。第二层结点 S 代表所有与 B 相似的已修复的缺陷报告，在这里，只要第一层和第二层对应的两个报告相似值不为 0，那么我们就认定它们相似并连接这两个结点。第三层结点 F 表示所有的代码源文件，如果第二层的缺陷报告在位于第三层的源文件中得到修复，那么我们就将结点 S 和结点 F 相连。然后我们借助式（10）来求解源代码文件 F 与新缺陷报告 B 的相似度。

$$SimiScore(B,F_j)=\sum_{所有与F_j相连的S_i}(Similarity(B,S_i)/n_i)$$

（10）

Si 代表所有与 Fj 相连的结点。ni 代表 Si 在第三层连接的结点总数。通过上式（5-10）的计算，我们得到了所有源文件的又一个关于新缺陷报告的相似值。

（3）综合各文件排名，定位可疑文件。

最后我们得到最终的相似值计算公式，如式（11）所示。

$$FinalScore=(1-\alpha)\times rVSMScore+\alpha\times SimiScore$$

（11）

其中 α 是权重因子并且 0≦α≦1。我们根据 FinalScore 对每个源文件进行降序排列得到所有可疑缺陷文件的排名。

3.1.3 安全缺陷报告识别的研究方案

本课题将排序学习和词嵌入模型相结合，提出了一种新方法来进行安全缺陷报告识别，该方法的基本流程图如图 3 所示。

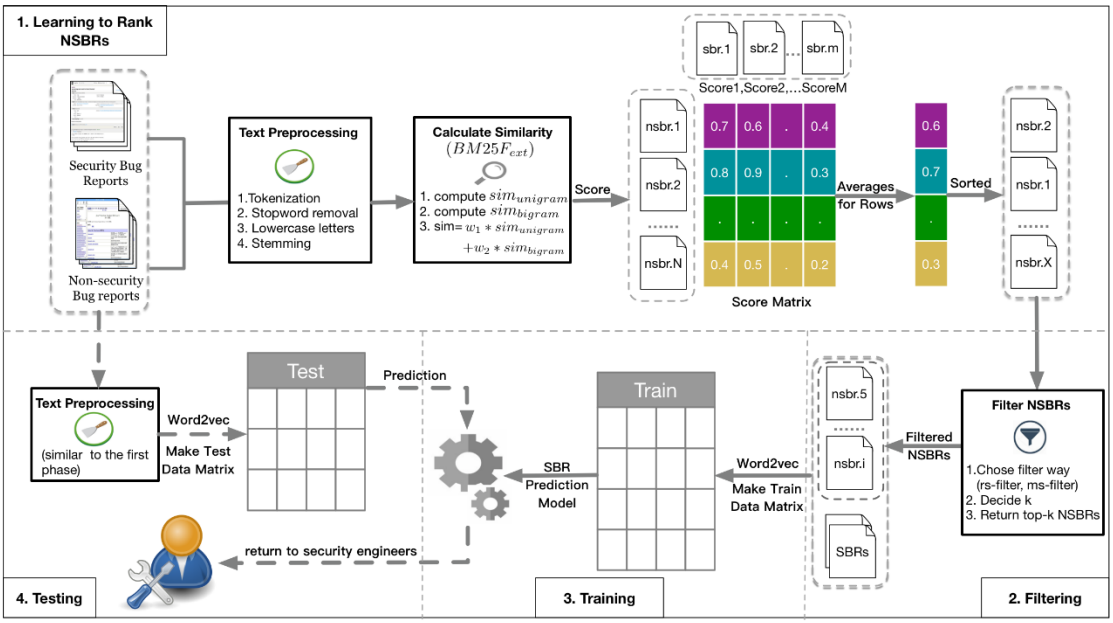

图 3 安全缺陷报告识别基本流程图

本方法共包含四个阶段：（1）排序学习阶段；（2）过滤阶段；（3）训练阶段；（4）测试阶段。下面对四个阶段进行详细的介绍：

（1）排序学习阶段。本阶段的目的在于将训练集中的非安全缺陷报告根据与安全缺陷报告的相似度进行排序，可分为三个步骤：

1）文本预处理：本步骤中我们对缺陷报告中的文本信息（即概要和描述）进行预处理，具体包括分词、去除停止词、将所有字母转换成小写以及词干提取。

2）计算相似度：我们利用 $BM25F_{ext}$ 算法[21]计算每一个非安全缺陷报告和安全缺陷报告的相似度，从而得到一个相似度矩阵，如图 3 所示。矩阵中的行和列分别对应非安全缺陷报告和安全缺陷报告。

不仅计算单个词在两个缺陷报告共同出现的情况下的相似度，还计算了两个连续的词共同出现在两个缺陷报告的情况下的相似度，再通过加权求和的方式得到两个缺陷报告的相似度，这种计算方式能够更充分地反映两个缺陷报告的相似程度。

3）非安全缺陷报告排序：本步骤中我们计算相似度矩阵中每一行元素的平均值，从而将二维的相似度矩阵压缩成一个向量，向量中的每一个元素代表一个非安全缺陷报告和所有安全缺陷报告的平均相似度。然后，我们根据向量中的平均相似度对所有非安全缺陷报告进行升序排序。

（2）过滤阶段。本阶段的目的是将与安全缺陷报告相似度较高的非安全缺陷报告过滤掉。本课题提出了两种过滤方式：倍数选择过滤和轮盘赌选择过滤。前者选取排序列表中的前 k 个非安全缺陷报告，后者根据轮盘赌算法选择非安全缺陷报告。轮盘赌算法根据一定的概率对非安全缺陷报告进行随机的选择，非安全缺陷报告与所有安全缺陷报告的不相似性越高，则被选择的概率越大。由于轮盘赌算法具有一定的随机性，因此能够覆盖更多可能的情况，但同时也增加了引入噪音的风险。需要注意的是，这两种过滤方式所保留的非安全缺陷报告的数量均设定为安全缺陷报告数量的一定倍数。

（3）训练阶段。本阶段我们将过滤后的非安全缺陷报告和所有的安全缺陷报

告作为训练集，并利用 word2vec 将缺陷报告中的每个词转换成低维稠密的向量表示，再将这些词向量的对应元素相加得到缺陷报告的向量表示。最后，我们分别用六种机器学习算法训练模型。

（4）测试阶段。将测试集中的每一个缺陷报告同样经过预处理并利用 word2vec 模型表示成低维稠密的向量，然后送入训练好的模型中，预测该缺陷报告所描述的缺陷是否为安全漏洞。

3.2 可行性分析

本项目以程序分析、机器学习、信息检索等技术和方法为理论基础，国内外在相关领域的理论研究成果可以为本项目所借鉴。

项目组成员近年来一直从事软件缺陷检测、软件故障定位相关理论和技术的研究工作：

（1）在故障定位方面：

1) 提出了一种基于加权软件行为图挖掘的软件错误定位方法，解决已有错误定位方法通常仅给出可疑语句排序而缺少必要的错误上下文信息，导致难于理解软件失效的产生原因的问题。
2) 提出了基于测试用例加权的方法和基于迭代预测的方法，以降低巧合正确性测试用例对软件错误定位精度的影响。
3) 提出了一种基于状态依赖概率建模的软件错误定位方法，提高了控制依赖相关的软件错误定位的精度。
4) 提出了一种基于联合依赖概率建模的软件错误定位方法，提高了数据依赖相关的软件错误定位的精度。

（2）在软件缺陷检测方面：

1) 提出了一种数据挖掘和程序静态分析相结合的克隆代码相关的软件缺陷的检测方法。
2) 提出基于控制结构的冗余代码缺陷检测方法，解决冗余代码缺陷检测复杂度较高且检测精度较低的问题。
3) 研究了并发缺陷暴露、检测与规避技术，动态检测和规避死锁缺陷和数据竞争缺陷，并在检测到缺陷的同时给出足够详细的现场信息，支持源码调试。

上述研究成果是本项目研究内容的前期研究工作基础，本项目是上述研究内容的扩展、深入和继续。

综上所述，我们的前期科研工作基础和取得的初步研究成果，可以确保本项目研究顺利进行以及研究目标可实现。

4. 本项目的特色与创新之处

（1）提出基于开发者社交网络的缺陷报告分派及故障定位方法。基于群体智能，挖掘开发人员之间的潜在合作关系，根据开发者对缺陷报告的评论建立开发者社交网络。使用机器学习的方法构建一个预测模型，通过该模型为每个新提交的缺陷报告分配一个相关的社区，推荐相关知识领域的合适的开发人员进行修复。并且综合领域知识和缺陷修复的历史信息定位软件故障，辅助软件调试、并优化回归测试范围。

（2）结合排序学习和词嵌入技术，提出了一个新的安全缺陷报告识别方法。该方法利用排序学习算法将与安全缺陷报告内容相似性较高的非安全缺陷报告过滤掉，而不是像之前的方法选择安全缺陷报告中 TF-IDF 值较高的词作为安全关键词并根据这些词来进行过滤。另一方面，我们使用词嵌入技术将缺陷报告表示成低维稠密向量，和之前使用的词袋模型的方法相比，这种分布式表示能够更好地表达缺陷报告所包含的语义信息，实现更好的预测效果。

5．年度研究计划及预期研究结果（包括拟组织的重要学术交流活动、国际合作与交流计划等）。

5.1 年度研究计划

（1）2019 年 1 月-2019 年 12 月：研究基于开发者社交网络的缺陷报告分派方法，进行算法设计、编程和实验，提交项目年度进展报告。

（2）2020 年 1 月-2020 年 12 月：研究基于缺陷报告的故障定位方法，进行算法设计、编程和实验，撰写研究论文，提交项目年度进展报告。

（3）2021 年 1 月-2021 年 12 月：研究安全缺陷报告识别方法，撰写研究论文，参加国际会议，提交项目年度进展报告。

（4）2022 年 1 月-2022 年 12 月：撰写研究论文，参加国际会议和交流，总结研究成果并提交项目总结报告。

5.2 预期研究结果

（1）构建基于开发者社交网络的缺陷报告分派模型和故障定位模型。

（2）在国内外重要学术刊物和会议上发表高水平研究论文 4 篇左右。

（3）申请发明专利、软件著作权 2-3 项。

（4）培养研究生 3-4 名。

。

（二）研究基础与工作条件

1．研究基础（与本项目相关的研究工作积累和已取得的研究工作成绩）；

（1）发表论文和专利授权情况

项目组成员近年来一直从事静态和动态程序分析以及智能软件相关理论和技术的研究，代表性论文见表 1-3。获国家发明专利授权 5 项，见表 4。

表 1 软件缺陷检测方向发表的研究论文

编号	论文题目	作者	期刊或会议名称
1	Using Reduced Execution Flow Graph to Identify Library Functions in Binary Code	*Qiu Jing、Su XiaoHong、Ma Peijun	IEEE Transactions on Software Engineering
2	Identifying functions in binary code with reverse extended control flow graphs	*Qiu, Jing、Su, Xiaohong、Ma, Peijun	Journal of Software-Evolution and Process
3	Distribution rule based bad smell detection and Refactoring scheme	Jiang, Dexun、*Ma, Peijun、Su, Xiaohong、Wang, Tiantian	Journal of Computational Information Systems,
4	Classes Bad Smell Detection and Refactoring schemes	Jiang Dexun、Ma Peijun、Su XiaoHong、Wang Tiantian	International Journal of Software Engineering & Applications
5	Distance metric based divergent change bad smell detection and refactoring scheme analysis	Jiang, Dexun、Ma, Peijun、Su, Xiaohong、Wang, Tiantian	International Journal of Innovative Computing Information and Control
6	Automatically Inferring Structure Correlated Variable Set For Concurrent Atomicity Safety	Pang Long、Su Xiaohong	International Journal of Software Engineering and Applications
7	并发缺陷暴露、检测与规避研究综述	苏小红、禹 振、王甜甜、马培军	计算机学报
8	Library functions identification in binary code by using graph isomorphism testings	*Qiu, Jing、Su, Xiaohong、Ma, Peijun	22nd IEEE International Conference on Software Analysis, Evolution, and Reengineering, SANER 2015
9	Identifying and understanding self-checksumming defenses in software	Qiu, Jing、Yadegari, Babak、Johannesmeyer, Brian、Debray, Saumya、Su, Xiaohong	5th ACM Conference on Data and Application Security and Privacy, CODASPY 2015

表 2 软件错误定位方向发表的研究论文

编号	论文题目	作者	期刊或会议名称
1	State dependency probabilistic model for fault localization	*Gong Dandan、Su Xiaohong、Wang Tiantian、Ma Peijun、Wang Yu	Information and Software Technology
2	A test-suite reduction approach to improving fault-localization effectiveness	*Gong Dandan、Wang Tiantian、Su Xiaohong、Ma Peijun	Computer Languages, Systems and Structures
3	结合用例约简与联合依赖概率建	苏小红、龚丹丹、王甜甜、	软件学报

	模的错误定位	马培军	
4	面向有效错误定位的测试用例优选方法	王克朝、王甜甜、苏小红、马培军、童志祥	计算机研究与发展
5	流敏感按需指针别名分析算法	逄龙、苏小红、马培军、赵玲玲	计算机研究与发展
6	软件错误自动定位关键科学问题及研究进展	王克朝、王甜甜、苏小红、马培军	计算机学报

表 3 在程序分析方向获得的国家发明专利授权

编号	发明专利名称	作者	专利号
1	一种程序代码编程模式著作权归属检测模型及著作权归属检测方法	王甜甜 王克朝, 苏小红 马培军	201210508663.5
2	基于状态依赖概率建模的软件错误定位方法	苏小红 龚丹丹 王甜甜 马培军	201310099998.0
3	基于联合依赖概率建模的软件错误定位方法	苏小红 龚丹丹 马培军 王甜甜	201310099997.6
4	一种使用逆向扩展控制流图的静态函数识别方法	邱景 苏小红 马培军 赵玲玲 王甜甜	2013102919410
5	一种使用收缩执行依赖图识别库函数的方法	邱景 苏小红 马培军 赵玲玲 王甜甜	2013105721740

（2）从事国家自然科学基金项目研究工作的基础

课题组成员苏小红自 2000 年以来，先后主持或参与完成多项国家自然科学基金项目的研究工作。

作为技术负责人，分别于 2002 年和 2005 年年底完成两项国家自然科学基金（批准号：69975005，60273083）的研究工作，排名第 2。

作为项目负责人，于 2009 年年底主持完成一项国家自然科学基金“基于程序转换和语义分析的编程题自动评分方法研究”（批准号：60673035，项目起止时间：2007 年 1 月-2009 年 12 月）的研究工作。研究成果已应用于 C 语言编程题考试自动评分系统中，该系统先后被国防科大、北京邮电大学、东北大学等 60 余所院校试用。在该项目中对程序静态分析、程序标准化、程序转换、程序相似度计算、程序匹配等关键技术进行了研究和探讨。

作为项目负责人，于 2011 年年底完成国家自然科学基金小额资助项目“数据挖掘和静态分析相结合的克隆代码缺陷检测及重构方法”（批准号：61073052，起止时间：2011 年 1 月-2011 年 12 月），重点研究基于数据挖掘和静态分析相结合的语法相似的克隆代码检测及相关缺陷检测方法。

作为项目负责人，于 2015 年年底完成国家自然科学基金资助项目“无定型克隆代码检测及重构方法”（批准号：61173021，起止时间：2012 年 1 月-2015 年 12

月），结合软件自动测试、程序静态分析、数据挖掘和模式聚类等理论，研究建立将克隆代码检测、相关缺陷检测和克隆代码重构有机融为一体的无定型克隆代码检测与重构模型。

（3）与国外大学联合培养博士研究生的工作基础

项目组与国外多所著名大学在程序分析领域联合培养博士生 4 人，如表 4 所示。

表 4 与国外大学联合培养博士研究生情况

姓名	大学	国别
逄龙	新加坡国立大学	新加坡
边奕心	美国马里兰大学巴尔的摩分校	美国
张凡龙	新加坡国立大学	新加坡
邱景	美国亚利桑那大学	美国
孙瑞	哈佛大学	美国

综上，申请人具有多年从事国家自然科学基金项目的研究工作经历，在与本项目相关的研究领域，具备扎实的理论基础和丰富的实践经验，并且取得了丰硕的研究成果，完全具备完成本项目的研究工作基础。

2. 工作条件（包括已具备的实验条件，尚缺少的实验条件和拟解决的途径，包括利用国家实验室、国家重点实验室和部门重点实验室等研究基地的计划与落实情况）；

申请者所在的哈尔滨工业大学计算机科学与技术学院智能软件技术研究中心，现配有设备如下：服务器 3 台，微机 50 余台，激光打印机 5 台。

以上设备和条件可以充分保证开发工作顺利进行。

3. 正在承担的与本项目相关的科研项目情况（申请人和项目组主要参与者正在承担的与本项目相关的科研项目情况，包括国家自然科学基金的项目和国家其他科技计划项目，要注明项目的名称和编号、经费来源、起止年月、与本项目的关系及负责的内容等）；

（三）其他需要说明的问题

1. 申请人同年申请不同类型的国家自然科学基金项目情况（列明同年申请的其他项目的项目类型、项目名称信息，并说明与本项目之间的区别与联系）。

无。

2. 具有高级专业技术职务（职称）的申请人或者主要参与者是否

存在同年申请或者参与申请国家自然科学基金项目的单位不一致的情况；如存在上述情况，列明所涉及人员的姓名，申请或参与申请的其他项目的项目类型、项目名称、单位名称、上述人员在该项目中是申请人还是参与者，并说明单位不一致原因。

无。

3. 具有高级专业技术职务（职称）的申请人或者主要参与者是否存在与正在承担的国家自然科学基金项目的单位不一致的情况；如存在上述情况，列明所涉及人员的姓名，正在承担项目的批准号、项目类型、项目名称、单位名称、起止年月，并说明单位不一致原因。

无。

4. 其他。