计算机科学
COMPUTER SCIENCE

基于开发者多元特征的软件缺陷自动分派方法

董夏磊, 项正龙, 吴泓润, 汪鼎文, 李元香

引用本文

董夏磊, 项正龙, 吴泓润, 汪鼎文, 李元香基于开发者多元特征的软件缺陷自动分派方法[J]. 计算机科学, 2022, 49(12): 81-88.

DONG Xia-lei, XIANG Zheng-long, WU Hong-run, WANG Ding-wen, LI Yuan-xiang. Automatic Assignment Method for Software Bug Based on Multivariate Features of Developers [J]. Computer Science, 2022, 49(12): 81-88.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.211100040

基于开发者多元特征的软件缺陷自动分派方法

董夏磊[1]　项正龙[2]　吴泓润[3]　汪鼎文[1]　李元香[1]
1 武汉大学计算机学院　武汉 430072
2 南京信息工程大学计算机与软件学院　南京 210044
3 闽南师范大学物理与信息工程学院　福建 漳州 363000
(xldong_whu@qq.com)

摘　要　软件缺陷修复是软件生命过程中一个不可忽视的问题,如何高效地进行软件缺陷的自动分派是一个十分重要的研究方向。目前已有的研究方法多侧重于缺陷报告的文本内容或开发者抛掷网络中的浅层信息,而忽视了开发者抛掷网络中的高层次拓扑信息。为此,提出了一个基于开发者多元特征的软件缺陷自动分派模型 MFD-GCN。该模型充分考虑开发者抛掷网络中的高层拓扑特征,并运用图卷积网络强大的网络特征提取能力,充分挖掘出代表开发者深层合作关系和修复偏好性的多元特征,并与缺陷报告文本特征一起训练分类器。模型在两个大型开源软件项目 Eclipse 和 Mozilla 上进行实验,实验结果表明,相比近年来提出的主流分派方法,MFD-GCN 模型在推荐前 K 个开发者时均取得了较好的推荐结果,其中,在 Eclipse 项目上 Top-1 推荐准确率达到了 69.8%,在 Mozilla 项目上达到了 59.7%。

关键词:自动分派;缺陷报告;开发者抛掷网络;图卷积网络;多元特征
中图法分类号　TP311.5

Automatic Assignment Method for Software Bug Based on Multivariate Features of Developers

DONG Xia-lei[1],XIANG Zheng-long[2],WU Hong-run[3],WANG Ding-wen[1] and LI Yuan-xiang[1]
1 School of Computer Science,Wuhan University,Wuhan 430072,China
2 School of Computer and Software,Nanjing University of Information Science and Technology,Nanjing 210044,China
3 School of Physics and Information Engineering,Minnan Normal University,Zhangzhou,Fujian 363000,China

Abstract　Software bug repair is a problem that cannot be ignored in the process of software life. How to efficiently assign software bugs automatically is a very important research direction. Now, the existing research methods mainly focus on the bug report's text content or the low-level information of the developers' tossing network, while ignoring the high-level topology information in the tossing network. Therefore, this paper proposes a software bug automatic assignment model MFD-GCN based on the developers' multivariate features. Model fully considers the high-level topological features in the developers' tossing network, and uses the powerful network feature extraction capabilities of graph convolution network to fully mine the multivariate features that represent developers' deep cooperation relationship and fixing preferences, and train the classifier together with the bug text features. The proposed method is evaluated on two large open-source software projects, i.e., Eclipse and Mozilla. Experimental results show that compared with the mainstream bug-assignment methods proposed in recent years, the MFD-GCN model has achieved state-of-art results in recommending the top K developers. The accuracy rate of top-1 recommendation on the Eclipse and Mozilla project reaches 69.8% and 59.7%, respectively.

Keywords　Automatic assignment, Bug report, Developers' tossing network, Graph convolution network, Multivariate feature

1　引言

近年来,以 GitHub 为代表的开源、开放的社群化软件开发环境的蓬勃发展,使得越来越多的公司接受并采用开源软件作为其 IT 战略的重要组成部分。然而,随着不同应用场景的开源软件的增加,软件缺陷也呈爆炸性激增,大量的软件缺陷被发现和提交到软件缺陷管理系统。众所周知,高效的软件缺陷修复是保障开源软件质量的重要环节。目前,大多数开源软件项目利用缺陷管理系统收集软件缺陷信息,并将软件缺陷分派给开发者以协助缺陷修复,这个过程叫做缺陷

到稿日期:2021-11-03　返修日期:2022-04-23
基金项目:国家自然科学基金(62106092,61672391)
This work was supported by the National Natural Science Foundation of China(62106092,61672391).
通信作者:吴泓润(dr.hongrunwu@gmail.com)

分派[1-2](Bug Assignment),也通常被称为开发者推荐。值得注意的是,很多缺陷经常被分派给不合适的开发者,导致软件缺陷需要被反复地重新分派新的开发者,造成时间和人力资源的浪费。文献[3-4]的统计分析表明,Eclipse 项目的一些缺陷甚至需要被分派数十次才能完成修复,而缺陷再分派平均耗时约 100 天。因此,高效的自动化缺陷分派方法对维护开源软件的安全和稳定十分重要。

为了提高缺陷的分派效率,许多自动化缺陷分派方法被提出。自动化缺陷分派指通过对历史缺陷数据进行统计分析学习等,自动为新出现的缺陷推荐合适的开发者来进行软件缺陷修复。从使用的缺陷数据的角度,现有的缺陷自动分派方法主要可分为 3 个方面:基于缺陷报告的文本内容[1,5-7],如利用元数据、摘要、描述和评论等文本信息来推荐开发者(元数据即缺陷报告的预定义字段);基于开发者之间的合作关系[4,8-9],如使用缺陷再分配图、开发者抛掷网络等来推荐开发者;以及基于缺陷报告文本内容和开发者合作关系的混合模型[10-12]。随着深度学习技术的日益成熟,一些深度学习模型也被用在缺陷分派问题中,例如运用卷积神经网络模型(Convolutional Neural Network,CNN)对缺陷报告文本进行学习并推荐开发者[13-14];运用循环神经网络(Recurrent Neural Network,RNN)对缺陷报告文本的句法、语义特征进行提取并推荐开发者[15];运用图卷积网络(Graph Convolutional Network,GCN)对缺陷报告中文本相似性关系进行学习并推荐开发者[16-17]。

然而,以上研究大多仅考虑缺陷报告的文本特征或开发者的浅层合作关系,很少涉及充分挖掘利用开发者抛掷网络中的高层次拓扑特征,如开发者的修复偏好性、深层合作关系等。针对此问题,本文提出了一个同时考虑开发者抛掷网络中的高层次拓扑特征和缺陷报告文本特征的分派模型(Multivariate Features of Developers-Graph Convolutional Network,MFD-GCN)。该模型首先根据缺陷报告中的修复历史数据、元数据等对开发者抛掷网络和节点属性进行建模,然后运用图卷积网络提取出代表开发者修复偏好和深层合作关系的多元特征,最后利用提取到的开发者多元特征和缺陷报告文本特征训练分类器,完成缺陷的自动分派。值得一提的是,模型在运用图卷积网络进行特征提取时,考虑了不同步长的邻居节点的作用,以获取开发者抛掷网络的高阶结构特征。本文在 Eclipse 和 Mozilla 两个常用的开源软件平台上进行实验[18],并将 MFD-GCN 模型与近年来提出的基于机器学习和深度学习的一些主流方法进行对比,实验结果表明 MFD-GCN 取得了较好的推荐效果。此外,实验结果还显示,开发者的多元特征对模型的推荐性能均有积极作用;同时,采用不同的融合函数也对模型性能有一定影响。

本文第 2 节介绍缺陷分派领域国内外学者的研究现状;第 3 节说明本文研究问题的预备知识;第 4 节介绍提出的 MFD-GCN 模型;第 5 节介绍实验数据并进行结果分析;最后总结全文。

2 相关工作

为了提高软件缺陷的修复效率,研究者提出了大量的缺陷自动分派方法。早期的研究方法主要是基于缺陷报告文本内容的机器学习分类方法,即从缺陷报告的文本内容(如摘要、描述和评论等)中提取文本特征并运用机器学习分类器实现缺陷的自动分派。例如,Xuan 等利用缺陷报告中的标题(Title)和描述(Description)部分的文本信息训练 SVM 和 MNB 等分类器完成开发者的推荐[5];Jonsson 等使用集成学习的思想,首先选取缺陷报告中不同字段的文本信息分别训练出不同的分类器,然后通过堆栈泛化(Stack Generalization)的方法集成不同分类器的分类结果,为缺陷推荐最合适的开发者[6]。与此同时,Jeong 等[4]和 Wang 等[8]发现,考虑开发者之间的合作关系能提高缺陷分派的效率,例如 Jeong 等利用马尔可夫链模型优化开发者之间的抛掷关系,从而提升缺陷分派的效率[4];Wang 等根据开发者对缺陷的修复历史、注释、评论等信息构造出开发者异质网络,通过构造的异质网络来分析多开发者之间的合作关系,并将分析结果用来辅助缺陷分派[8]。随后,有研究发现基于缺陷文本和开发者关系的混合方法能进一步提升缺陷分派的准确率[10-12]。例如,Shi 等首先根据缺陷报告之间的文本相似度,找出相似度高的缺陷报告所对应的修复者,然后根据开发者在再分配图中的依赖关系生成预测再分配路径[10];Bhattacharya 等则通过使用附加属性、多特征抛掷图等技术来提升推荐效果[11-12]。

随着深度学习技术的发展,一些学者开始运用深度学习的技术提取缺陷报告的文本特征。例如 Mani 等提出了一种基于深度双向循环神经网络(DBRNN-A)的缺陷自动分派模型,该模型采用无监督的方式从缺陷报告文本中学习句法和语义特征,并使用注意力机制关注文本的重要部分以提高模型的推荐效果[15];Xi 等提出的 ITriage 模型首先采用序列到序列模型(Sequence-to-Sequence Model)来联合学习缺陷报告文本和抛掷序列的特征,然后使用缺陷报告文本、元数据和抛掷序列的特征训练分类器[19]。

近年来,随着图神经网络(Graph Neural Networks,GNN)在非欧氏空间数据上的成功应用,一些基于图神经网络的缺陷分派模型逐渐被提出[16-17]。例如,Zaidi 等首先根据缺陷报告文本中单词-单词和单词-文档之间是否同时出现来建立缺陷报告关系网络,然后运用图卷积网络从缺陷报告网络中提取特征,最后用提取到的特征训练分类器[16];与 Zaidi 等不同,Li 等提出基于缺陷报告文本之间的余弦相似性来建立缺陷报告关系网络,然后从建立的网络中提取特征来训练分类器[17]。以上两个模型均取得了较好的推荐结果。

3 预备知识

3.1 缺陷报告

很多大型开源软件项目往往使用缺陷管理系统来收集和追踪软件缺陷,例如 Eclipse 和 Mozilla 两个开源软件均使用 Bugzilla 系统进行管理[18]。软件缺陷在管理系统中以缺陷

报告的形式存在。一个缺陷报告主要包括：摘要（Summary）、预定义字段（Pre-defined Filed）、修复历史（History）、描述（Description）和评论（Comment）5 个主要部分，如图 1 所示。其中，摘要、描述和预定义字段包含了丰富的缺陷报告基础信息；评论部分是开发者对该缺陷的修复建议；而修复历史则蕴含了开发者之间的合作交互信息[20]。

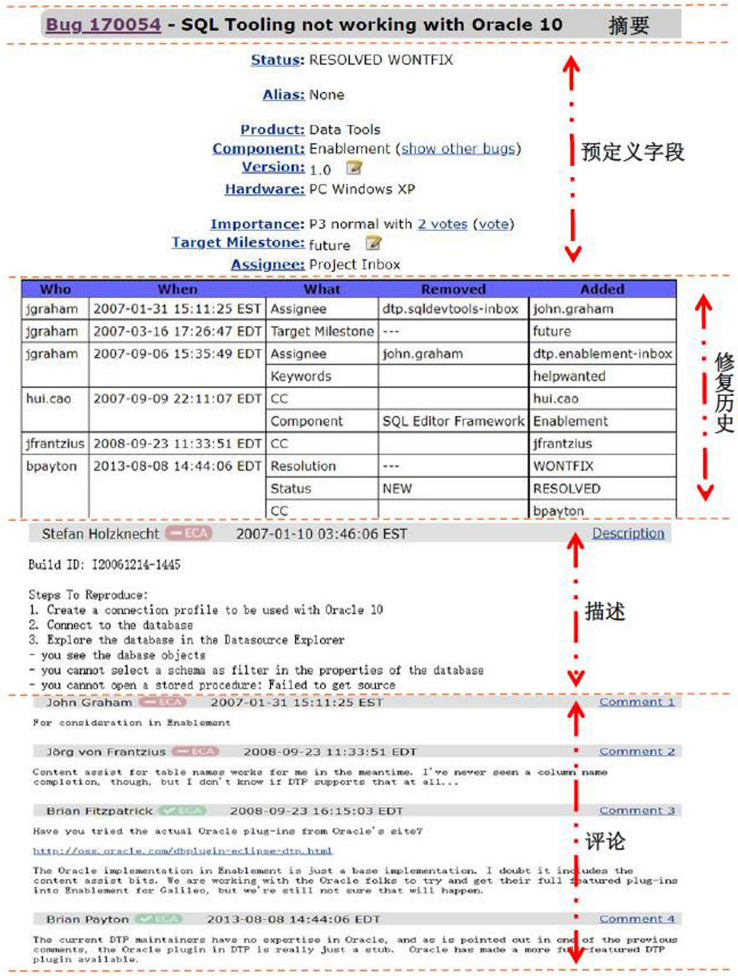

图 1　Eclipse 项目中 ID 为 170054 的缺陷报告

Fig. 1　Bug report with ID 170054 in Eclipse project

一个缺陷报告被提交到缺陷追踪系统后，会经历分派、抛掷（又称“再分配”）、解决、验证、关闭等几个状态，各状态构成的生命周期如图 2 所示。一个新的缺陷报告（New）被提出后，管理员会根据缺陷报告的内容将缺陷分派（Assigned）给合适的开发者，如果该开发者没有完成，该缺陷将继续被抛掷（Tossed）给下一个可能修复该缺陷的开发者，缺陷报告被持续抛掷直到最终被修复（Resolved），最终经过验证（Verified）后关闭（Closed）[1]。当然，一些无效的、重复的、或者无法被修复的缺陷也会被标记为修复状态。本文的研究主要关注缺陷的分派状态，即如何为软件缺陷推荐最合适的开发者。

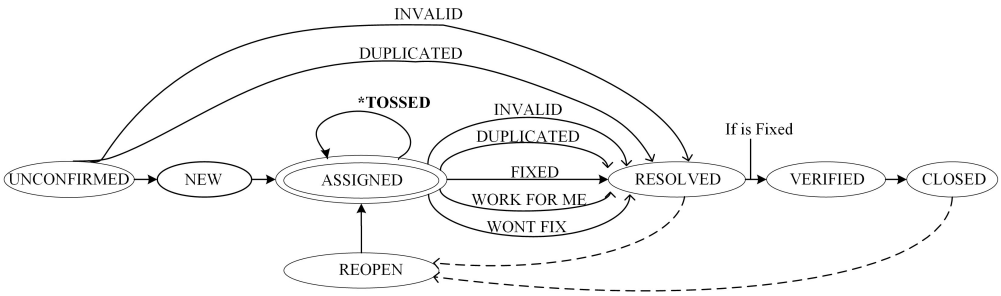

图 2　缺陷报告的生命周期

Fig. 2　Life cycle of bug report

3.2　开发者抛掷网络

缺陷报告的修复历史记录中包含了开发者之间的抛掷关系，根据开发者之间的抛掷关系可以构建开发者抛掷网络 $G(V,E)$，其中 $V=\{d_1,d_2\cdots,d_C\}$ 表示开发者集合，C 为开发者数量。由于开发者之间存在不同强弱程度的信任关系，本文根据开发者之间的抛掷次数，将 G 的邻接矩阵 $\boldsymbol{A}$ 定义为式(1)。

$$\boldsymbol{A}_{ij}=\begin{cases}1, & \text{if } count(d_i,d_j)\geqslant\tau\\ 0, & \text{otherwise}\end{cases} \tag{1}$$

其中，若开发者d_i和d_j之间的抛掷次数等于或超过τ次，则$\boldsymbol{A}_{ij}=1$，否则$\boldsymbol{A}_{ij}=0$。

开发者修复过的软件缺陷通常可以反映其在缺陷修复中的偏好，例如经常选择修复某一主题的缺陷报告。本文考虑开发者在缺陷修复过程中的偏好性，并将其作为开发者属性，用于缺陷报告的分派。根据 Wu 等前期的研究[21]，开发者的工作主题(Working Topic)和工作组件(Working Component)两种属性对软件缺陷的修复影响最大，因此本文继续采用工作主题和工作组件作为开发者属性。

开发者的工作主题指开发者修复的大多数缺陷报告所属的主题，根据开发者曾经参与修复的缺陷报告的主题来定义。我们将开发人员d_i偏好于修复主题t的缺陷报告的概率定义为式(2)：

$$P(d_i,t)=\frac{\sum\limits_{b\in B_{d_i}}\theta(b,t)}{\sum\limits_{t\in T}\sum\limits_{b\in B_{d_i}}\theta(b,t)} \tag{2}$$

其中$\theta(b,t)$是由 LDA 模型(Latent Dirichlet Allocation)模型[22]生成的主题词，B_{d_i}是开发者d_i曾参与修复的缺陷报告集。

开发者可能至少有一个工作主题，因为任何一个缺陷报告可能有若干个主题。根据式(2)中开发者d_i偏好的修复主题，我们将开发者d_i的工作主题$\hat{t}$定义为式(3)：

$$d_i[\hat{t}]=\arg\max_{t\in t^*}P(d_i,t) \tag{3}$$

其中，t^*是d_i偏好的修复主题集合。

开发者的工作组件根据开发者曾经参与修复的缺陷报告的所属组件来确定。开发者d_i偏好修复某一组件c的缺陷的概率如式(4)所示：

$$P(d_i,c)=\frac{\sum\limits_{b\in B_{d_i}}\delta(b,c)}{|B_{d_i}|} \tag{4}$$

其中，B_{d_i}是开发者d_i曾参与修复的缺陷报告集，$\delta(b,c)$表示缺陷报告 b 是否属于组件c。根据式(4)，开发者d_i的工作组件可定义为式(5)：

$$d_i[\hat{c}]=\arg\max_{c\in c^*}P(d_i,c) \tag{5}$$

其中，c^*表示开发者d_i曾经参与修复缺陷的所有组件。由于一个开发者可能会修复许多缺陷，这些缺陷可能来自不同的组件。因此，开发者可以偏好于修复多个组件的缺陷报告。

结合式(2)－式(5)，我们将开发者抛掷网络G上的节点属性记为$\boldsymbol{X}=\{x_{d_1},x_{d_2}\cdots,x_{d_c}\}$，其中$x_{d_i}$表示开发者$d_i$的工作主题$\hat{t}$或工作组件$\hat{c}$。当其为工作主题时，$x_{d_i}$表示$\hat{t}$；当其为工作组件时，$x_{d_i}$表示$\hat{c}$。

3.3 图卷积网络

图卷积网络(Graph Convolutional Network，GCN)由 Kipf 等[23]于 2016 年第一次提出使用，其被广泛用于知识图谱和社交网络等网络状结构数据中[24]。图卷积网络借助卷积操作对局部结构的感知能力及网络中普遍存在的节点与邻居节点的相似关系，通过聚合邻居节点信息来更新自身节点的信息，从而达到提取网络空间拓扑特征的目的。本文运用图卷积网络对开发者抛掷网络进行特征提取，充分提取出开发者的多元特征，用于下游的推荐任务。

GCN 的层特征传播操作和卷积过程如式(6)所示：

$$H^{l+1}=f(H^l,\boldsymbol{A})=\sigma(\hat{\boldsymbol{A}}H^l\boldsymbol{W}^l) \tag{6}$$

其中，H^l表示该层的特征输入，若为H^0，则为原始数据输入；$\boldsymbol{A}$表示网络的邻接矩阵；$f(\cdot)$表示非线性函数，一般为激励函数。具体而言，$\hat{\boldsymbol{A}}$表示正则化后的拉普拉斯矩阵，即$\hat{\boldsymbol{A}}=\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$，其中$\boldsymbol{D}$为邻接矩阵$\boldsymbol{A}$的度矩阵，$\boldsymbol{W}^l$表示被训练学习的权重参数，$\sigma$函数为非线性变换函数，一般为 Relu，Tanh 等激励函数。GCN 模型一般有多层，通过多层卷积操作，完成对网络空间拓扑特征的提取。

4 MFD-GCN 模型

本文提出的 MFD-GCN 模型框架如图 3 所示。主要包括 3 个模块：模块 I 对缺陷报告文本内容进行特征提取；模块 II 运用图卷积网络从开发者抛掷网络中提取多元特征；模块 III 使用缺陷报告的文本特征与开发者的多元特征训练分类器，实现缺陷的自动分派。

4.1 缺陷报告的文本特征提取

模块 I 实现对缺陷报告的文本特征的提取，输入为缺陷报告的 TF-IDF 值矩阵$\boldsymbol{X}_{tf}$。本文计算$\boldsymbol{X}_{tf}$使用的缺陷报告文本内容包括摘要、描述、评论 3 部分，具体操作为：首先对缺陷报告文本数据进行分词、去停词、词干抽取等预处理，然后采用 TF-IDF 模型计算出每个缺陷报告的 TF-IDF 值向量$\boldsymbol{x}$，再将值向量$\boldsymbol{x}$作为缺陷报告的特征向量组成$\boldsymbol{X}_{tf}$，其中$\boldsymbol{X}_{tf}\in\mathbb{R}^{N\times m}$，$N$为缺陷报告数量，$m$为特征向量维度。进一步，采用一个全连接层(Full Connected Layer)将$\boldsymbol{X}_{tf}$转换到新的特征空间$\boldsymbol{B}$(Bug Feature)中，具体计算过程如式(7)所示：

$$\boldsymbol{B}=\boldsymbol{X}_{tf}\boldsymbol{W}_{fc} \tag{7}$$

其中，$\boldsymbol{B}\in\mathbb{R}^{N\times d}$，$\boldsymbol{W}_{fc}$表示全连接层中被训练学习的参数矩阵，$d$为转换后的特征空间维度。

4.2 开发者的多元特征提取

模块 II 通过图卷积网络对开发者抛掷网络提取多元特征。其输入为：开发者抛掷图$G(V,E)$，开发者的工作主题属性矩阵$\boldsymbol{X}_t$和工作组件属性矩阵$\boldsymbol{X}_c$。

根据 Abu-El-Haija 等学者的研究，通过对不同阶的邻居节点信息进行叠加能有效提高分类效果[25]，例如直接(一阶)邻居节点往往包含网络的局部结构信息，而二阶或三阶等远距离邻居节点则倾向于蕴含更高层次的拓扑信息。因此，对不同阶的邻居节点进行信息叠加能有效地捕获网络的局部以及全局的拓扑特征。式(6)中 GCN 对网络的一阶邻居进行卷积，即对正则化后拉普拉斯矩阵$\hat{\boldsymbol{A}}$进行操作。$\hat{\boldsymbol{A}}^k$是对$\hat{\boldsymbol{A}}$的k次幂计算，其蕴含节点随机游走k步的拓扑信息，从而达到使用浅层 GCN 获取高层次拓扑特征的目的。在本文中，$k=1$表示开发者的直接合作关系，$k>1$表示开发者的深层合作关系。

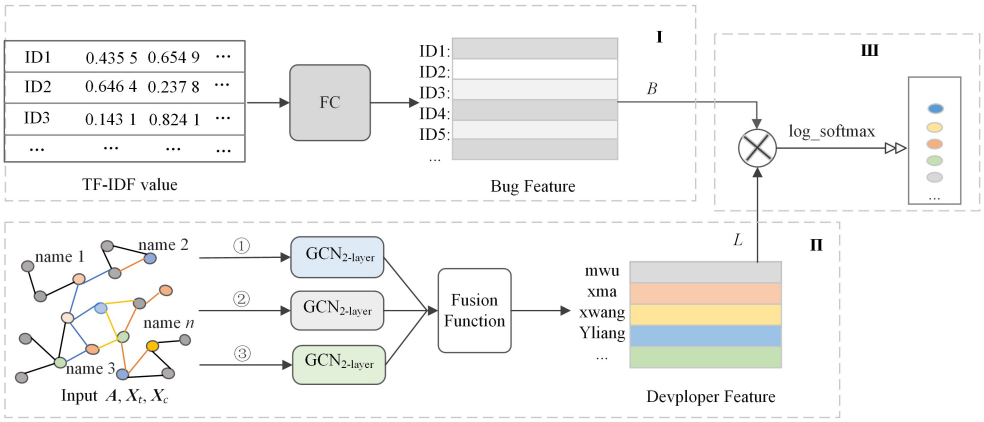

图 3 MFD-GCN 模型示意图

Fig. 3 Framework of MFD-GCN model

因此，我们分别组合节点在不同步长（$k=1$ 和 $k=2$）下的信息，同时还组合开发者的不同属性。具体如图 3 所示。模型对开发者抛掷网络提取多元特征主要通过 3 个通道来完成，通道一根据式(8)提取特征，考虑节点随机游走 1 步的拓扑信息，开发者属性选择$\boldsymbol{X}_t$，提取的开发者局部拓扑特征记为$\boldsymbol{H}_t$；通道二根据式(9)提取特征，考虑节点随机游走两步的拓扑信息，开发者属性选择$\boldsymbol{X}_t$，提取的开发者高层拓扑特征记为$\boldsymbol{H}_{t^2}$；通道三根据式(10)提取特征，考虑节点随机游走两步的拓扑信息，开发者属性选择$\boldsymbol{X}_c$，提取的开发者高层拓扑特征记为$\boldsymbol{H}_{c^2}$。式(8)—式(10)中，σ 为 Relu 激励函数，$\boldsymbol{W}_t^{l_t}$ 为网络训练参数，$\boldsymbol{H}\in\mathbb{R}^{C\times d}$，$d$ 为特征向量维度，与 4.1 节中缺陷报告的特征维度大小保持一致。

$$\boldsymbol{H}_t=\sigma(\hat{\boldsymbol{A}}\cdot\sigma(\hat{\boldsymbol{A}}\boldsymbol{X}_t\boldsymbol{W}_0^{l_0})\boldsymbol{W}_0^{l_1}) \tag{8}$$

$$\boldsymbol{H}_{t^2}=\sigma(\hat{\boldsymbol{A}}^2\cdot\sigma(\hat{\boldsymbol{A}}^2\boldsymbol{X}_t\boldsymbol{W}_1^{l_0})W_1^{l_1}) \tag{9}$$

$$\boldsymbol{H}_{c^2}=\sigma(\hat{\boldsymbol{A}}^2\cdot\sigma(\hat{\boldsymbol{A}}^2\boldsymbol{X}_c\boldsymbol{W}_2^{l_0})\boldsymbol{W}_2^{l_1}) \tag{10}$$

最后，根据式(11)融合 3 种特征$\boldsymbol{H}_t$，$\boldsymbol{H}_{t^2}$，$\boldsymbol{H}_{c^2}$，得到开发者最终的特征矩阵 $\boldsymbol{L}$，$\boldsymbol{L}\in\mathbb{R}^{C\times d}$。其中融合函数$\theta_i$可选用的包括最大化(max)、叠加(add)、累积(mul)，即$\theta_i\in\{\max,\text{add},\text{mul}\}$。

$$\boldsymbol{L}=\boldsymbol{H}_t\cdot\theta_1\cdot\boldsymbol{H}_{t^2}\cdot\theta_2\cdot\boldsymbol{H}_{c^2} \tag{11}$$

4.3 特征融合与分类

模块 III 根据模块 I 提取的缺陷报告文本特征与模块 II 提取的开发者多元特征训练分类器。具体定义如式(12)所示。其中分类函数采用 log_softmax 函数以避免计算过程数据溢出，同时也加快了计算速度，y'为预测标签，即所推荐的开发者。

$$y'=\log\_\text{softmax}(\boldsymbol{B}\cdot\boldsymbol{L}^{\mathrm{T}}) \tag{12}$$

模型采用有监督的训练方式，运用历史缺陷报告数据进行训练。由于开发者推荐问题为多类别分类任务，因此 MFD-GCN 采用交叉熵作为损失函数。定义如式(13)所示：

$$Loss=-\sum_{c=1}^{C}y_c\log y_c' \tag{13}$$

其中，C 为所有开发者的数量，y 为真实标签，y'为式(12)得到的预测标签。

5 评估实验及结果分析

5.1 实验数据集

本文实验数据来自大型开源软件项目 Eclipse 和 Mozilla 中的缺陷报告[18]，两个项目均使用 Bugzilla 系统来管理缺陷信息。在 Bugzilla 系统中，每个软件缺陷都有一个半结构化的缺陷报告，缺陷报告如图 1 所示。本文选取状态更改信息标记为“已验证”和“已修复”的缺陷报告作为实验数据[1)]。

数据集的具体信息如表 1 所列，Eclipse 项目由在 2001-10—2011-10 期间已验证修复的约 9.5 万个缺陷报告组成，其中共包含 5 个产品（Product）、121 个组件（Component）和 1 420 名开发者（Developer）；Mozilla 数据集由在 1998-04—2011-04 期间已验证修复的约 11.8 万个缺陷报告组成，其中共包含 5 个产品、197 个组件和 3 634 名开发者。

表 1 实验数据集

Table 1 Experimental dataset

Project	Bug Report	Product	Component	Deve-loper
Eclipse	95 053	5	121	1 420
Mozilla	117 781	5	197	3 634

5.2 实验设置

实验中的 MFD-GCN 模型基于深度学习库 PyTorch Geometric 实现[2)]，实验在配备 Intel（R） Core i9 10920X 和 NVIDIA GEFORCE RTX3080 的服务器上完成。模型训练中学习率设置为 0.001，Dropout 设置为 0.5，最大迭代周期为 40 000，τ 设置为 2，缺陷报告特征矩阵$\boldsymbol{X}_{tf}$的维度 m 设置为 2 000，$\boldsymbol{W}_{fc}$维度 d 设置为 300，式(11)中的特征融合函数θ_1默认设置为 add 操作，θ_2默认设置为 mul 操作，采用 Adam 优化器进行模型训练。缺陷分派问题本质为推荐问题，与当前研究常用的评价指标一致。本文所有实验均采用推荐准确率（Accuracy）指标来评价模型的好坏。

本文共设计了 3 组实验，分别从 MFD-GCN 与基准方法的推荐结果对比、各个通道多元特征对模型性能的贡献以及多元特征组合方式对模型性能的影响 3 个方面分析验证

1) https://github.com/ssea-lab/BugTriage/

2) https://pytorch-geometric.readthedocs.io/en/latest/

MFD-GCN 的推荐性能。

实验 1(5.3.1 节) 对比了当前主流方法与本文提出的 MFD-GCN 方法在 Eclipse 与 Mozilla 数据集上的推荐准确率。主流方法包括近年来缺陷分派领域的 5 种模型,其中,BOW+SVM,TF+SVM,DF+SVM[21]为基于机器学习的方法,DBRNN-A[15]和 ITriage[19]为基于深度学习的方法。实验中训练集比例分别设置为 0.4,0.6,0.8,实验记录 Top-K 准确率,其中 K 取 1,3,5。

BOW+SVM 方法:该方法通过 BOW(Bag of Words)模型描述缺陷报告的文本特征,然后训练 SVM 分类器以推荐合适的开发者。实验中该方法的词频阈值范围为 30～3000,SVM 分类器取默认参数。

TF+SVM 方法:该方法先采用 TF-IDF 模型计算出缺陷报告文本的 TF-IDF 值,再根据 TF-IDF 值训练 SVM 分类器,其中缺陷报告文本的 TF-IDF 值与模块 I 定义保持一致,SVM 分类器取默认参数。

DF+SVM 方法:该方法考虑开发者属性信息,使用开发者修复的历史缺陷报告中的关键字字段的信息来训练 SVM 分类器[21],SVM 分类器取默认参数。

DBRNN-A 方法:该方法基于深度双向循环神经网络(Bi-directional RNN)模型建立,模型采取无监督的方式从缺陷报告文本中学习句法和语义特征,并通过注意力机制来提高推荐效果[15],模型参数与原文保持一致。

ITriage 方法:该方法首先使用序列到序列模型学习缺陷报告文本和抛掷序列的特征,然后使用分类模型将缺陷报告的文本内容、元数据和抛掷序列的特征进行整合,以训练分类器[19],模型参数与原文保持一致。

实验 2(5.3.2 节) 探究 MFD-GCN 中各个通道提取的多元特征对模型性能的贡献。实验分别考虑开发者的局部特征$\boldsymbol{H}_t$、高层拓扑特征$\boldsymbol{H}_{t^2}$和$\boldsymbol{H}_{c^2}$对模型的推荐性能的影响,特征融合函数θ_1和θ_2均取 add 操作,训练集设置为 0.8,记录 Top-K 的准确率,其中 K 取 1,3,5。

实验 3(5.3.3 节) 探究 MFD-GCN 中的融合函数对模型推荐准确率的影响。特征融合函数θ_1和θ_2分别选择 mul,add,max 中的一种,其他实验设置保持一致,实验数据训练集设置为 0.8,记录 Top-K 的准确率,其中 K 取 1,3,5。

5.3 实验结果

5.3.1 MFD-GCN 与主流方法的推荐结果对比

表 2 列出了 MFD-GCN 和 5 种主流方法在完整的 Eclipse 和 Mozilla 数据集上的推荐准确率。由表中可以看出,随着训练样本比例的增加,各方法在 Eclipse 和 Mozilla 项目上的推荐准确率 Top-1,Top-3,Top-5 均逐渐提升。从两个数据集的测试结果可以发现,在基于机器学习方法上,第二组以 TF-IDF 值作为缺陷报告的文本信息在训练集比例为 0.8 情况下,Eclipse 和 Mozilla 项目上 Top-1 推荐效果分别可达到 54.3%和 46.1%,远超过直接使用词频信息的 BOW+SVM 组,而第三组 DF+SVM 经过属性字段挑选后的文本信息比未挑选过属性字段的第一组和第二组效果要好,Eclipse 项目 Top-1 推荐最高准确率可达64.5%,Top-5 可达 83.2%,而 Mozilla 项目 Top-1 推荐最高准确率可达 48.6%,Top-5 可达 68.1%;同时,从前 3 组实验结果可以看出,使用 TF-IDF 值作为缺陷报告文本信息比直接用词频信息推荐效果要好,而使用挑选过的属性字段作为缺陷报告文本信息则效果更好。

表 2 MFD-GCN 和对比实验推荐准确率

Table 2 Recommendation accuracy of MFD-GCN and comparative methods

	Approach	Top 1/3/5		
		40%	60%	80%
Eclipse	BOW+SVM	15.7/28.4/37.1	19.2/33.6/42.5	24.1/36.9/47.2
	TF+SVM	47.1/60.5/69.7	50.0/62.3/71.6	54.3/65.0/73.1
	DF+SVM	52.2/62.7/72.3	60.3/69.8/75.9	64.5/72.4/83.2
	DBRNN-A	17.7/27.9/31.5	21.9/30.1/34.7	23.6/33.1/37.0
	ITriage	30.5/41.1/52.4	37.3/55.1/66.4	47.5/70.2/77.3
	MFD-GCN	**66.8/83.5/87.6**	**68.9/85.3/89.1**	**69.8/85.4/89.4**
Mozilla	BOW+SVM	12.4/23.9/31.2	16.2/28.8/37.3	21.0/31.5/40.6
	TF+SVM	37.2/50.7/59.3	40.4/54.0/62.8	46.1/59.2/66.7
	DF+SVM	40.3/53.1/61.6	44.4/60.2/65.4	48.6/62.7/68.1
	DBRNN-A	12.9/20.5/24.6	18.7/29.1/35.5	21.2/32.2/39.5
	ITriage	25.6/41.5/50.3	35.0/47.6/57.2	45.1/63.2/69.9
	MFD-GCN	**58.1/73.5/77.7**	**59.3/74.5/78.7**	**59.7/75.0/79.2**

在深度学习方法上,MFD-GCN 的推荐准确率比未使用开发者信息的 DBRNN-A 和 ITriage 两个模型均有较大的提升,相比机器学习方法也有较大的提升。在各训练集比例下,Eclipse 项目中 Top-1 推荐准确率提升效果均在 5%以上,最高可达 69.8%,Top-3 和 Top-5 提升效果也均在 6%以上;而在 Mozilla 项目中,Top-1 提升效果均在 11%以上,最高可达 59.7%,Top-3 和 Top-5 提升效果也均在 9%以上,并且随着推荐人数的增加,推荐效果也越好。其中,在 Eclipse 项目上,Top-5 准确率最高达到了 89.4%,而在 Mozilla 项目上最高达到了 79.2%。与主流方法相比,MFD-GCN 模型的推荐性能提升明显,验证了 MFD-GCN 的有效性。

5.3.2 MFD-GCN 中开发者多元特征对模型性能的影响

在 Eclipse 和 Mozilla 数据集上,MFD-GCN 各个通道提取的多元特征对模型性能的影响结果如图 4 所示。其中 Ga 组只考虑开发者的局部特征$\boldsymbol{H}_t$,Gb 组考虑开发者的局部特征$\boldsymbol{H}_t$和高层拓扑特征$\boldsymbol{H}_{t^2}$,Gc 组考虑开发者的局部特征$\boldsymbol{H}_t$和高层拓扑特征$\boldsymbol{H}_{c^2}$,Gd 组考虑开发者的局部特征$\boldsymbol{H}_t$和高层拓扑特征$\boldsymbol{H}_{t^2}$和$\boldsymbol{H}_{c^2}$。如图 4 所示,在 Eclipse 数据集上,仅使用开发者局部特征的 Ga 组的 Top-1 推荐准确率比其他同时使用开发者高层拓扑特征的组别低 0.4%～2.5%,Top-3 和 Top-5 推荐准确率最大差值为 1.2%和 1.3%,而使用开发者两种高层拓扑特征的 Gd 组比使用一种高层拓扑特征的 Gb 组和 Gc 组的推荐性能在 K 取 1,3,5 的情况下均高 1%左右;对于 Mozilla 数据集,使用开发者局部特征和高层拓扑特征的 Gd 组比仅使用局部特征的 Ga 组的 Top-1 推荐准确率高出 2.6%,比使用一种高层拓扑特征的 Gb 组和 Gc 组的 Top-3 与 Top-5 推荐准确率高出 1.1%～2.2%不等。由此可见,本文提取的开发者的多元特征对模型推荐效果均有积极作用;同时,从开发者抛掷网络中提取的高层拓扑特征能有效提升模型的推荐性能。

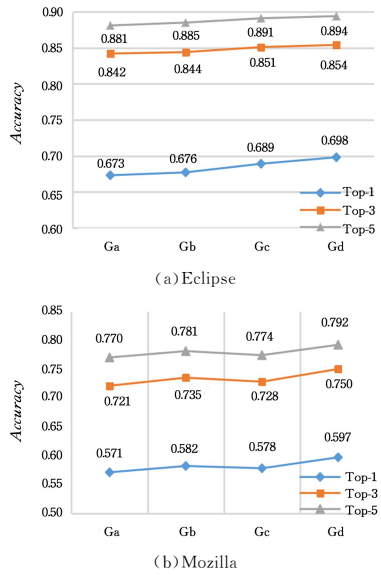

图 4 不同的特征组合对模型性能的影响

Fig. 4 Influence of different feature combinations on model performance

5.3.3 MFD-GCN 中融合函数对模型性能的影响

表 3 列出了 MFD-GCN 在 Eclipse 和 Mozilla 数据集上不同融合函数θ_1和θ_2组合下模型的 Top-K 的推荐准确率，其中“+”代表 add 函数，“*”代表 mul 函数。

如表 3 所列，不同的融合函数组合对模型推荐性能也会产生一定的影响。对于 Eclipse 数据集，采取$(\boldsymbol{H}_t+\boldsymbol{H}_t^2)*\boldsymbol{H}_c^2$融合函数组合时效果最好，而$\theta_1$和$\theta_2$全取 mul 的效果最差；对于 Mozilla 数据集，采取$\boldsymbol{H}_t+\boldsymbol{H}_t^2*\boldsymbol{H}_c^2$，$\boldsymbol{H}_t*\boldsymbol{H}_t^2+\boldsymbol{H}_c^2$和$\max(\boldsymbol{H}_t*\boldsymbol{H}_t^2,\boldsymbol{H}_c^2)$融合函数组合时效果较好，同样地，$\theta_1$和$\theta_2$全取 mul 的效果最差。从总体的结果可以看出，融合函数选择 add，max 的组合更有利于开发者多元特征的融合，而全部使用 mul 函数的融合效果较差。

表 3 不同的融合函数组合对模型性能的影响

Table 3 Influence of different combination of fusion functions on model performance

θ_1，θ_2	Eclipse			Mozilla		
	Top-1	Top-3	Top-5	Top-1	Top-3	Top-5
+，+	68.4	85.1	89.0	58.3	73.5	78.0
，	65.5	81.4	85.6	52.3	65.6	69.7
+，*	**69.8**	85.4	89.4	59.7	**75.0**	**79.2**
*，+	69.2	85.6	89.5	**60.1**	**75.0**	78.9
(+)*	69.7	**86.1**	**89.9**	59.9	73.5	78.0
max(*，)	68.4	85.1	89.0	59.9	**75.0**	**79.2**
max(，)*	69.1	85.8	89.7	60.0	74.7	79.1

结束语 本文从开发者抛掷网络入手，提出了一个端到端的软件缺陷自动分派模型 MFD-GCN，该模型充分考虑了开发者抛掷网络中的高层次拓扑特征对推荐性能的影响，通过图卷积网络提取到开发者的多元特征，并与缺陷报告文本特征一起，实现了缺陷的自动分派，最后在开源项目 Eclipse 和 Mozilla 上进行实验，验证了 MFD-GCN 模型的有效性。

在实验过程中我们发现，考虑开发者抛掷网络的高层次拓扑信息能够有效提高模型的推荐效果，而本文考虑的高层次拓扑信息也仅仅考虑了开发者的主题属性和组件属性，尚未挖掘到开发者的其他特征属性，这将是未来工作的方向；并且，对于开发者多元特征的融合方式，本文仅考虑简单的融合函数，未来将进一步考虑将注意力机制加入特征融合过程中，以取得更优的推荐性能。

参 考 文 献

[1] ANVIK J. Automating Bug Report Assignment[C]// Proceedings of the 28th International Conference on Software Engineering. New York: ACM Press, 2006: 937-940.

[2] ANVIK J, HIEW L, MURPHY G C. Who Should Fix This Bug? [C]// Proceedings of the 28th International Conference on Software Engineering. New York: ACM Press, 2006: 361-370.

[3] Inc. STATISTA. Projected Revenue of Open Source Software from 2008 to 2020[EB/OL]. https://www.statista.com/statistics/270805/projected-revenue-of-open-source-software-since-2008/.

[4] JEONG G, KIM S, ZIMMERMANN T. Improving Bug Triage with Bug Tossing Graphs[C]// Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering. New York: ACM Press, 2009: 111-120.

[5] XUAN J F, JIANG H, HU Y, et al. Towards Effective Bug Triage with Software Data Reduction Techniques[J]. IEEE Transactions on Knowledge and Data Engineering, 2014, 27(1): 264-280.

[6] JONSSON L, BORG M, BROMAN D, et al. Automated Bug Assignment: Ensemble-based Machine Learning in Large Scale Industrial Contexts[J]. Empirical Software Engineering, 2016, 21(4): 1533-1578.

[7] SHI X W, MA Y T. Software Bug Triaging Method Based on Text Classification and Developer Rating[J]. Computer Science, 2018, 45(11): 193-198.

[8] WANG S, ZHANG W, YANG Y, et al. DevNet: Exploring Developer Collaboration in Heterogen-eous Networks of Bug Repositories[C]// Processing of the 7th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. New York: IEEE Press, 2013: 193-202.

[9] WU W, ZHANG W, YANG Y, et al. DREX: Developer Recommendation with K-Nearest Neighbor Search and Expertise Ranking[C]// Processing of the 18th Asia-Pacific Software Engineering Conference. New York: IEEE Press, 2011: 389-396.

[10] SHI G X, ZHAO F Y. Software Defect Assignment Method Based on Defect Similarity and Tossing Graph[J]. Computer Science, 2016, 43(11): 246-251.

[11] BHATTACHARYA P, NEAMTIU I. Fine-grained Incremental Learning and Multi-feature Tossing Graphs to Improve Bug Triaging[C]// 2010 IEEE International Conference on Software Maintenance. New York: IEEE Press, 2010: 1-10.

[12] BHATTACHARYA P, NEAMTIU I, SHELTON C R. Automated, Highly-accurate, Bug Assignment Using Machine Learning and Tossing Graphs[J]. Journal of Systems and Software, 2012, 85(10): 2275-2292.

[13] SONG H Z, MA Y T. DeepTriage: An Automatic Triage Method for Software Bugs Using Deep Learning[J]. Journal of

Chinese Computer Systems, 2019, 40(1): 128-134.
[14] GUO S, ZHANG X, YANG X, et al. Developer Activity Motivated Bug Triaging: via Convolutional Neural Network[J]. Neural Processing Letters, 2020, 51(3): 2589-2606.
[15] MANI S, SANKARAN A, ARALIKATTE R. DeepTriage: Exploring the Effectiveness of Deep Learning for Bug Triaging [C]// Proceedings of the ACM India Joint International Conference on Data Science and Management of Data. New York: ACM Press, 2019: 171-179.
[16] ZAIDI S F A, LEE C G. Learning Graph Representation of Bug Reports to Triage Bugs using Graph Convolution Network [C]// 2021 International Conference on Information Networking (ICOIN). New York: IEEE Press, 2021: 504-507.
[17] LI Y X, DONG X L, XIANG Z L, et al. A Graph Convolutional Neural Network Based Approach for Software Bug Triage[J]. Journal of Wuhan University (Natural Science Edition), 2020, 66(3): 244-252.
[18] LAMKANFI A, PEREZ J, DEMEYER S. The Eclipse and Mozilla Defect Tracking Dataset: a Genuine Dataset for Mining Bug Information [C]// the 10th Working Conference on Mining Software Repositories (MSR). New York: IEEE Press, 2013: 203-206.
[19] XI S Q, YAO Y, XIAO X S, et al. Bug Triaging Based on Tossing Sequence Modeling[J]. Journal of Computer Science and Technology, 2019, 34(5): 942-956.
[20] ZHANG J, WANG X, HAO D, et al. A Survey on Bug-report Analysis[J]. Science China Information Sciences, 2015, 58(2): 1-24.
[21] WU H, LIU H, MA Y. Empirical Study on Developer Factors Affecting Tossing Path Length of Bug Reports[J]. IET Software, 2018, 12(3): 258-270.
[22] BLEI D M, NG A Y, JORDAN M I. Latent Dirichlet Allocation [J]. Journal of Machine Learning Research, 2003, 3: 993-1022.
[23] KIPF T N, WELLING M. Semi-supervised Classification with Graph Convolutional Networks [J]. arXiv: 1609.02907, 2016.
[24] YING R, HE R, CHEN K, et al. Graph Convolutional Neural Networks for Web-scale Recommender Systems [C]// Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. New York: ACM Press, 2018: 974-983.
[25] ABU-EL-HAIJA S, KAPOOR A, PEROZZI B, et al. N-GCN: Multi-scale Graph Convolution for Semi-supervised Node Classification[C]// Proceedings of Machine Learning Research. MLR Press, 2020: 841-851.

DONG Xia-lei, born in 1998, postgraduate. His main research interests include deep learning and intelligent software.

WU Hong-run, born in 1989, Ph.D, associate professor, is a member of China Computer Federation. Her main research interests include complex networks and graph neural network.

(责任编辑: 何杨)