学　　号2017201318

密　　级

哈尔滨工程大学本科生毕业论文

自动生成缺陷报告优先级的自动预测方法

学院（系）名 称：计算机科学与技术学院

专 业 名 称：软件工程

学 生 姓 名：谈尤帅

指 导 教 师：刘刚 副教授

哈尔滨工程大学

2021年6月

**自动生成缺陷报告优先级的自动预测方法 谈尤帅 哈尔滨工程大学**

学　　号2017201318

密　　级

自动生成缺陷报告优先级的自动预测方法

Automatic prediction approach for generating bug report priority automatically

**学生姓名：**谈尤帅

**所在学院：**计算机科学与技术学院

**所在专业：**软件工程

**指导教师：**刘刚

**职称：**副教授

**所在单位：**计算机科学与技术学院

**论文提交日期：**2021年6月

**论文答辩日期：**2021年6月

**学位授予单位：**哈尔滨工程大学

摘 要

随着软件中缺陷的增加，缺陷追踪系统被广泛地应用，用户发现缺陷后可以撰写缺陷报告上传至缺陷追踪系统。为了使重要的缺陷优先被修复，开发者会给每个缺陷报告标注出优先级。但是人工标注优先级是一件费时费力的工作，所以研究者们提出一些自动标注优先级的算法。不过现存算法是存在着不足的，比如忽略了长距离与非连续语义信息、未能很好地解决数据不平衡问题、需要大量的训练数据等等。

为了更好地实现缺陷报告优先级的自动生成，本文提出一个基于图卷积神经网络并使用加权损失函数的算法。算法首先提取缺陷报告中的部分文本作为每个缺陷报告的数据，并根据所有的数据构建出一个文本异构图。图包含着两类结点，一类是文本结点，一类是词结点，每个文本结点都对应着一个优先级标签。然后算法利用图卷积神经网络对图进行结点分类以实现缺陷报告优先级的自动生成。为了解决数据不平衡问题，算法在模型训练中使用了加权损失函数。

为了评估算法，本文在四个开源项目Mozilla、Eclipse、Netbeans和GCC的缺陷报告中进行实验。实验结果表明本文提出的算法显著地超过两个基线算法，降低了数据不平衡对结果的影响，并且在低训练集比例的情况下也能达到预期的效果。

关键词：图卷积神经网络；缺陷报告；优先级自动生成；训练集比例；数据不平衡

**ABSTRACT**

With the increasing number of software bugs, bug tracking systems have been widely used. Users could write bug reports and upload them to the system when they find bugs. To make sure that significant bugs can be fixed preferentially, developers should suggest a priority level for a bug report. However, manual priority assignment is a time-consuming and arduous work. To automatically predict the priority levels, recent researchers have proposed some approaches. However, these approaches still contain several deficiencies that include ignoring long-distance and nonconsecutive semantics, not solving the problem of imbalanced data, and needing much training data.

To bridge this gap, in this paper, the author proposes a novel algorithm based on graph convolutional networks. This algorithm first extracts several textual parts of bug reports as data. Then it builds a heterogeneous graph which has two kinds of nodes. One kind is bug report which has a certain priority level, the other is word. At last, it applies graph convolutional networks to classify nodes in this graph to realize priority automatically generating. To solve the problem of imbalanced data, it constructs a weighted loss function in the training phrase.

To evaluate our algorithm, the author conducts the priority prediction on four open-source projects, including Mozilla, Eclipse, Netbeans, and GCC. Experimental results show that it significantly surpasses the two baseline algorithms, reduces the impact of imbalanced data on the results, and could still perform well with a small amount of training data.

**Keywords:** Graph Convolutional Network; Bug Report; Automatically Generating Priority; Ratio of Training Set; Unbalanced Data

目 录

[第1章 绪论 1](#_Toc74895911)

[1.1 论文的研究背景 1](#_Toc74895912)

[1.2 国内外研究现状 1](#_Toc74895913)

[1.2.1 优先级自动生成 2](#_Toc74895914)

[1.2.2 严重度自动预测 3](#_Toc74895915)

[1.2.3 现存研究的不足 4](#_Toc74895916)

[1.3 论文的研究目的和主要工作 4](#_Toc74895917)

[1.4 论文组织结构 5](#_Toc74895918)

[第2章 论文相关概念和技术 7](#_Toc74895919)

[2.1 缺陷追踪系统 7](#_Toc74895920)

[2.1.1 缺陷报告 7](#_Toc74895921)

[2.1.2 缺陷修复 8](#_Toc74895922)

[2.2 文本分类算法 10](#_Toc74895923)

[2.2.1 传统文本分类算法 10](#_Toc74895924)

[2.2.2 深度文本分类算法 11](#_Toc74895925)

[2.3 图神经网络 13](#_Toc74895926)

[2.3.1 图神经网络的背景和用途 13](#_Toc74895927)

[2.3.2 图神经网络的分类 14](#_Toc74895928)

[2.3.3 图神经网络的未来工作 15](#_Toc74895929)

[2.4 本章小结 16](#_Toc74895930)

[第3章 基于图卷积神经网络的缺陷报告优先级自动生成 17](#_Toc74895931)

[3.1 PPWGCN算法概述 17](#_Toc74895932)

[3.2 数据预处理 19](#_Toc74895933)

[3.3 文本异构图构建 20](#_Toc74895934)

[3.4 基于图卷积神经网络的文本分类 21](#_Toc74895935)

[3.5 本章小结 23](#_Toc74895936)

[第4章 实验设计与分析 25](#_Toc74895937)

[4.1 实验设置 25](#_Toc74895938)

[4.1.1 评价指标 25](#_Toc74895939)

[4.1.2 实验数据 25](#_Toc74895940)

[4.1.3 参数设置 27](#_Toc74895941)

[4.2 PPWGCN与两个基线算法的效果的对比 27](#_Toc74895942)

[4.2.1 动机与实验步骤 27](#_Toc74895943)

[4.2.2 结果与分析 27](#_Toc74895944)

[4.3 PPWGCN在低训练集比例时的效果评估 32](#_Toc74895945)

[4.3.1 动机与实验步骤 32](#_Toc74895946)

[4.3.2 结果与分析 32](#_Toc74895947)

[4.4 加权损失函数解决数据不平衡问题的验证 38](#_Toc74895948)

[4.4.1 动机与实验步骤 38](#_Toc74895949)

[4.4.2 结果与分析 38](#_Toc74895950)

[4.5 本章小结 42](#_Toc74895951)

[结 论 43](#_Toc74895952)

[参考文献 44](#_Toc74895953)

[攻读学士学位期间发表的论文和取得的科研成果 47](#_Toc74895954)

[致 谢 48](#_Toc74895955)

# 第1章 绪论

## 1.1 论文的研究背景

缺陷修复在软件维护中扮演着一个非常重要的角色。但是软件行业的蓬勃发展导致大量缺陷的出现，这使得缺陷修复成为一项费时且费力的工作。例如，调查发现从2021年1月1日至2021年3月8日超过10000份关于Mozilla项目的缺陷报告被提交给开发者。这意味着Mozilla项目平均每天会收到超过150份缺陷报告。为了及时定位缺陷以加快软件的更新工作，很多大型项目的开发团队开始使用缺陷追踪系统（比如Bugzilla和Jira）管理缺陷。当用户发现某个缺陷的时候，他可以撰写缺陷报告上传到缺陷追踪系统中，然后开发者会根据这些缺陷报告修复相应的缺陷。因此，越来越多的缺陷报告被提交到缺陷追踪系统中。

由于每个开源项目的开发人员是有限的，所以当过多的缺陷报告被提交至缺陷追踪系统中之后就会出现缺陷报告积压的现象。这会进而导致很多亟待修复的缺陷得不到及时的修复，而很多不严重的缺陷却很快得到重视。为了解决这个问题，开发者先为每个缺陷报告标注优先级，然后优先级高的缺陷报告优先被修复。在实际的应用场景中，开发者首先根据缺陷报告判断缺陷报告中包含的缺陷是增强（Enhancement）或者是问题（Problem）。如果这个缺陷属于增强，便会被搁置。而如果这个缺陷属于问题，那么开发者就会对相应的缺陷报告进行优先级标注。以Bugzilla系统为例，缺陷报告的优先级从P1到P5被分为五种，P1代表最高的优先级，P5表示最低的优先级。P3一般是开发者觉得不算特别严重和不严重且难以判断为P2或P4的一类标签，所以这类标签的缺陷报告非常多。尽管人工标注缺陷报告优先级可以获得不错的效果，但是这样比较费时且需要很多的人力。所以研究者们致力于构建出能够高效地实现缺陷报告优先级自动生成的算法。

## 1.2 国内外研究现状

近年来，研究者们设计出一些基于机器学习的分类算法以实现缺陷报告优先级的自动生成，本节将介绍这些算法。而在开发者标注缺陷报告优先级之前，用户会在缺陷报告中标注出其严重度，这被作为优先级预测的参考内容。为了帮助用户给缺陷报告标注严重度，研究者也提出了一些自动标注缺陷报告严重度的算法。因为严重度预测与优先级标注类似，所以本节也会介绍缺陷报告严重度预测的现存研究。

### 1.2.1 优先级自动生成

Tian等人[1]提出了一种基于线性回归（Linear Regression）的分类方法（DRONE）以实现缺陷报告优先级的自动生成。作者的方法可以分为两个部分，第一个部分是特征提取，第二个部分是分类模块。为了更好地提取出缺陷报告中的信息，作者在特征提取部分考虑了很多维度，比如时间、缺陷报告的作者、严重度等等。作者根据不同维度设计出不同的编码方式最终得到每个缺陷报告的特征向量。在分类模块中，作者首先使用线性回归去捕捉缺陷报告的特征向量与优先级的关系。但是因为数据非常的不平衡（Imbalanced），所以作者利用爬山法找到四个阈值以实现分类算法。作者使用了Eclipse中的缺陷报告作为数据，Eclipse是一个可以支持软件开发多个方面的集成开发环境。作者选择了一些缺陷报告严重度预测算法作为基线算法，实验结果表明作者的算法在平均F-measure这个指标上超过基线算法209%。

Alenezi和Banitaan[2]尝试使用三种机器学习分类算法以实现缺陷报告优先级的自动生成，它们分别为朴素贝叶斯分类器（Naïve Bayes Classifier），决策树分类器（Decision Trees Classifier）和随机森林分类器（Random Forests Classifier）。朴素贝叶斯是一个基于概率的分类算法，它假设所有的特征都是独立的。它使用贝叶斯定理在给定一组特征值的情况下找到概率最大的标签。决策树是一种基于树结构的分类器。决策树是一种预测模型，可以根据现有数据的各种属性值来确定新样本的依存值。树中的每个内部节点代表一个属性，而叶节点则表示一个类标签。决策树通过从树的根结点遍历至叶节点来对每个实例进行分类。随机森林是一种集成学习方法，它可以在模型训练中生成多个决策树，每棵树都对应着一个类标签。决策树输出的类标签由单个树输出的类别的众数决定。作者提取缺陷报告的两类特征作为数据。第一类特征是缺陷报告中的description部分，由于文本是非结构化的数据，所以作者对description进行了预处理。作者首先分别对description进行词干化，去停用词，去标点，去空格和去数字。然后作者使用词袋模型向量化每个缺陷报告的description部分。第二类特征是缺陷报告中的元数据（Meta-data），作者选择了三个部分：构件（Component），操作系统（Operating System）和严重度（Severity）。其中，构件代表了这个缺陷所在的构件，操作系统代表了这个缺陷被发现时所在的操作系统，严重度代表了这个缺陷报告的严重度。作者的实验结果表明决策树和随机森林的效果比朴素贝叶斯好，而且元数据更加重要。

Umer等人[3]提出了一种基于卷积神经网络（Convolutional Neural Network）的方法（cPur）实现自动生成缺陷报告的优先级。作者选择缺陷报告中的description作为数据，为了减少模型耗费的计算资源，首先对description进行预处理。预处理分为五个步骤，第一步是词干化，作者同时去掉了特殊字符、标点符号和空格。第二步是去除停用词，因为不是所有的词都会对结果精度的提高有帮助。第三步是纠正拼写，因为缺陷报告是由个人撰写且无纠错机制所以较容易出现拼错的单词。第四步是提取出否定词和修饰词，因为加上否定词或修饰词会使语义发生很大的改变。第五步是词形还原，同时也把文本全部转化为小写。预处理后，作者开始提取每个description蕴含的情感倾向。当用户遇到非常严重的缺陷后很有可能会非常激动，所以情感倾向可以作为一个判断缺陷报告优先级的有效指标。作者使用情感分析工具得到每个description的情感倾向是正面（Positive）的或负面（Negative）的。得到description的情感倾向之后，作者直接把情感词加入到预处理得到的词中以获得最终的数据。为了更好地从数据中提取出语法和语义信息，作者利用word2vec模型得到文本的嵌入（Embedding）。Word2vec是一个单隐藏层的神经网络，它是一个学习高密度连续词表征的高效方法。模型分为CBOW（Continuous Bag-of-Words）和Skip-Gram。CBOW的目标函数是根据前后词预测中间的词，而Skip-Gram与之相反，其目标函数是根据某个词预测前后的词。根据经验Skip-Gram的效果会更好，所以作者使用基于Skip-Gram的word2vec模型获得数据的嵌入。最后作者使用卷积神经网络构建出一个分类器以实现缺陷报告优先级标签的自动生成。相对于循环神经网络（Recurrent Neural Network），卷积神经网络处理长距离依赖的能力更强且无梯度爆炸的风险。通过使用不同大小的卷积核，卷积神经网络可以同时提取长距离依赖和短距离依赖信息。最终作者使用Tian等人[1]的数据进行实验，结果表明他们的算法优于基线算法。

Abdelmoez等人[4]提出一个利用朴素贝叶斯分类器预测缺陷报告优先级的方法。他们的数据来自三个大型开源项目。他们根据平均修复时间判断缺陷报告的优先级。

Choudhary等人[5]提出一个基于支持向量机的缺陷报告优先级预测模型，他们提取缺陷发生的频率作为特征。

### 1.2.2 严重度自动预测

缺陷报告的严重度与优先级是两个相似的部分。严重度是由撰写缺陷报告的用户提供的而优先级是由开发者标注的。根据调研发现，研究严重度自动预测的论文比优先级自动生成的多很多。

Menzies和Marcus[6]是第一个研究缺陷报告严重度自动预测的团队。他们的研究数据是来自美国航空航天局（NASA）提供的缺陷报告。作者首先把缺陷报告中的description部分切割为独立的词，然后将这些词词干化并去掉其中的停用词。最终作者利用名为RIPPER的算法训练这些数据。

Lamkanfi等人[7]是第一个尝试在开源项目的缺陷报告中进行严重度预测的团队，他们的工作是Menzies和Marcus工作的延伸。他们只考虑缺陷报告严重度中的五类标签，分别为blocker，minor，critical，major和trivial。其中blocker，critical和major属于严重的标签，minor和trivial属于不严重的标签。

Tan等人[8]提出一个基于逻辑回归的轻量级算法，这项研究是关于严重度预测最近的一次研究。他们首先从Stack Overflow问答网站爬取关于缺陷的内容，然后使用BM25算法扩充缺陷报告的内容。

刘文杰[9]提出一个特征序列重构方法，将数据特征子集的维度降低至50%。然后他使用集成特征选择的算法，优化现存特征选择算法，最终提高了严重度预测的精度。

Sabor等人[10]提出一个基于堆栈记录和分类特征的严重度预测模型。他们将堆栈记录相似度与分类特征相似度线性的结合以预测缺陷报告的严重度。

### 1.2.3 现存研究的不足

尽管研究者们提出了一些算法实现缺陷报告优先级的自动生成，但是这些方法仍然是存在着一些不足的。

Tian等人[1]提出的DRONE算法仍然不能很好地解决数据不平衡问题，因为那些数量少的标签的预测效果不好。不仅如此，可能因为传统的机器学习算法并不能很完备地提取出数据中的特征，DRONE得到的结果精度较低。

Umer等人[3]提出的cPur算法是基于卷积神经网络的，虽然卷积神经网络擅长提取局部连续词中的语义信息，但是它忽略了全局共现词中的长距离和非连续语义信息。这就导致Umer取得的结果精度较低，而且Umer在应对不平衡数据时的表现非常差，很多时候只能预测出一个标签。

## 1.3 论文的研究目的和主要工作

为了弥补现存方法中的不足，作者提出了一个基于图卷积神经网络的算法，修改Yao等人[11]提出的文本分类算法作为算法的主体部分。算法考虑了缺陷报告中的五个部分作为数据，他们分别为severity，product，component，description和summary。然后使用自然语言处理（Natural Language Processing）技术对数据进行预处理并利用预处理后的数据构建出一个包含词结点和文本结点的异构图。通过上述的步骤，缺陷报告优先级自动生成被转化为结点分类问题。然后为了获得全局共现词的信息以得到令人满意的效果，作者使用图卷积神经网络为异构图建模。因为图神经网络可以使标签信息和全局共现词信息在整张图中传播，所以方法可以充分利用数据的全局信息。为了解决数据不平衡的问题，作者通过引入标签罚项构建出一个新的加权交叉熵损失函数。

Yao等人[11]的实验结果表明图卷积神经网络的结点分类在少量训练数据的情况下也能获得不错的实验结果。软件工程社区是一个较为小众的社区，所以软件工程社区相对于人工智能社区缺乏优质的标注数据，比如ImageNet包含了千万级别张标注图像。不仅如此，软件工程社区的文本信息是具有时效性的，比如多年前的缺陷报告会因为版本的迭代而失去指导意义。所以软件工程社区非常需要利用少量训练数据也能获得不错实验结果的算法，而图卷积神经网络的这个特点就使之很适合被用于软件工程领域的任务之中。

为了验证新的算法，作者选择两个最新且较优秀的算法DRONE[1]和cPur[3]作为基线算法，并在四个开源项目Mozilla、Eclipse、Netbeans和GCC的缺陷报告上进行实验。

本文的主要工作如下：

（1）爬取四个开源项目Mozilla、Eclipse、GCC和Netbeans中的缺陷报告作为缺陷报告优先级自动生成算法的实验数据。

（2）复现两个基线算法DRONE和cPur，DRONE利用线性回归并训练出四个阈值以实现缺陷报告优先级的自动生成；cPur结合情感分析使用卷积神经网络实现缺陷报告优先级的自动生成。

（3）设计出一个基于图卷积神经网络并设置加权损失函数的缺陷报告优先级自动生成算法。进行三个实验：算法与两个基线算法的对比；评估算法在低训练集比例下的效果是否变化不大；验证算法中的加权损失函数是否可以降低不平衡数据对实验结果的影响。

## 1.4 论文组织结构

本文旨在设计出一种用于实现缺陷报告优先级自动生成的算法，算法可以应对数据不平衡问题，在低训练集比例的情况下也能有不错的效果。围绕这个中心，论文的组织结构如下：

第1章是绪论。首先对论文的研究背景和国内外研究现状进行阐述，主要介绍了优先级自动生成和严重度自动预测。随后介绍现存研究中的不足以及论文的研究目的和主要工作，最后给出论文的组织结构。

第2章是论文相关概念和技术。分别介绍了缺陷追踪系统、文本分类算法和图神经网络。因为本文的课题是缺陷追踪系统中的一部分；本文提出的算法的本质是一个文本分类算法；算法是基于图卷积神经网络的。

第3章是对本文提出的算法的详细描述。分别介绍了预处理、构建文本异构图和搭建图卷积神经网络的过程。

第4章是实验设计与分析。首先介绍了实验的设置，然后从动机、具体步骤和结果分析三个方面阐述设计的三个实验。这三个实验分别为：把算法与基线算法作对比；验证算法在低训练集比例的情况下是否也能表现良好；验证算法的加权损失函数能否应对数据不平衡问题。

# 第2章 论文相关概念和技术

本文旨在设计出一个用于自动化标注缺陷报告优先级的算法。本章将会介绍本文涉及到的概念和技术，包括缺陷追踪系统，文本分类算法和图神经网络。

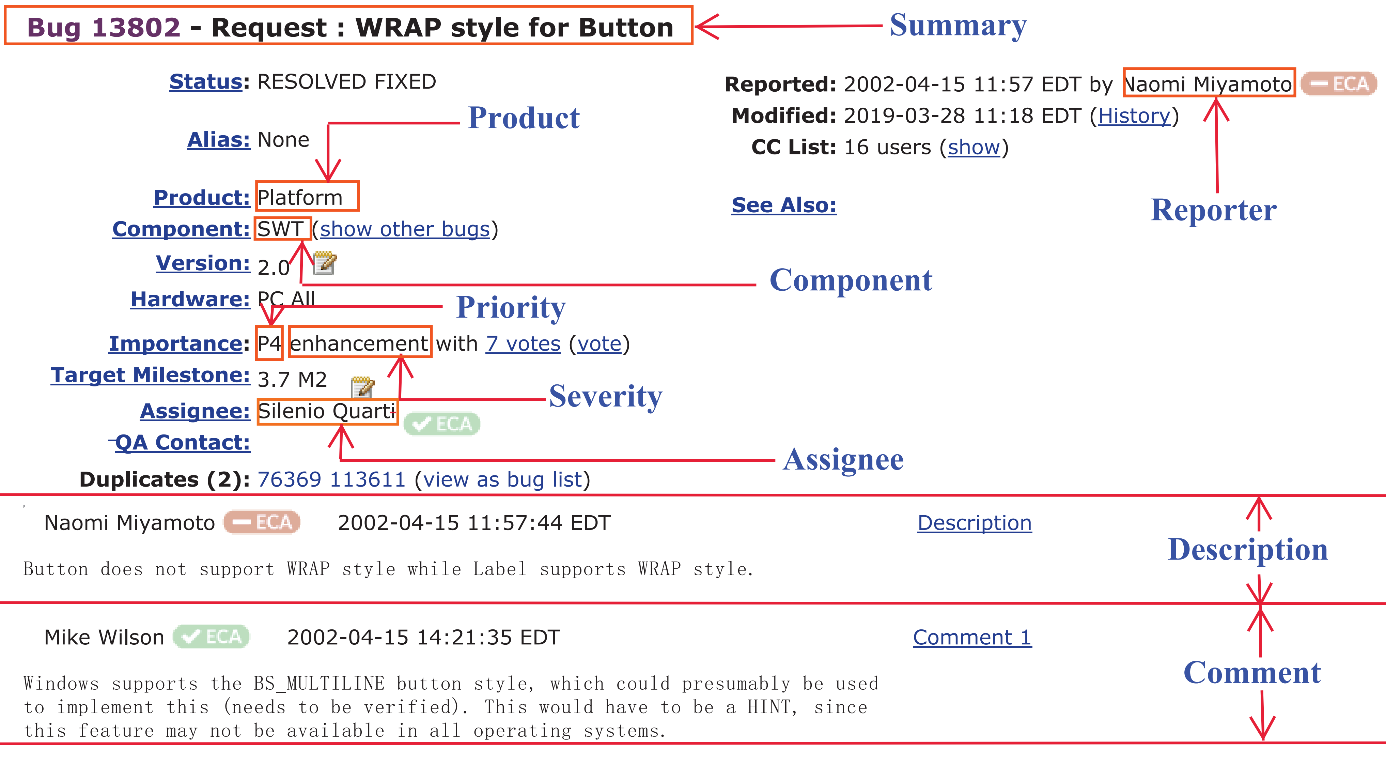
## 2.1 缺陷追踪系统

缺陷报告是缺陷追踪系统用来管理缺陷的工具，本节首先将介绍缺陷报告的各个部分，然后描述从缺陷报告被提交到缺陷被修复的整个流程。

### 2.1.1 缺陷报告

缺陷报告是开发者和用户之间的媒介，也是项目开发人员间沟通合作的基础。缺陷报告首先由用户提交至系统中，项目相关的开发人员和测试人员都可以查看所有的缺陷报告，权限高的负责人甚至可以对其内容进行修改[12]。

Bugzilla和Jira是最常用的两个缺陷追踪系统[13]，本文中的实验数据也是出自于Bugzilla。图2.1展示了一个Bugzilla系统中Eclipse项目的缺陷报告实例。

图2.1 Eclipse缺陷报告示例

作者标出了缺陷报告中的主要内容，它们分别是：

* 摘要（Summary）简要地介绍缺陷；
* 产品（Product）：被缺陷影响到的产品；
* 报告人（Reporter）：撰写缺陷报告的人；
* 构件（Component）：该缺陷干扰到的构件；
* 优先级（Priority）：开发者判定的缺陷的缓急程度；
* 严重度（Severity）：用户判定的缺陷的缓急程度；
* 受托人（Assignee）：负责修复缺陷的开发者；
* 描述（Description）：对缺陷的详细描述；
* 评论（Comment）：任意开发者对修复缺陷的建议。

其中只有报告人和受托人信息是姓名信息，考虑到数据是来自开源项目而会有很多的开发者参与到缺陷修复中，因而姓名信息不具有明显的特征，所以作者不把报告人和受托人这两个部分纳入数据中。而评论中可能包含着很多与这个缺陷无关的信息，所以也不考虑这个部分。综上所述，最终提取缺陷报告中的五个部分作为数据，它们分别为：摘要、产品、构件、描述和严重度。

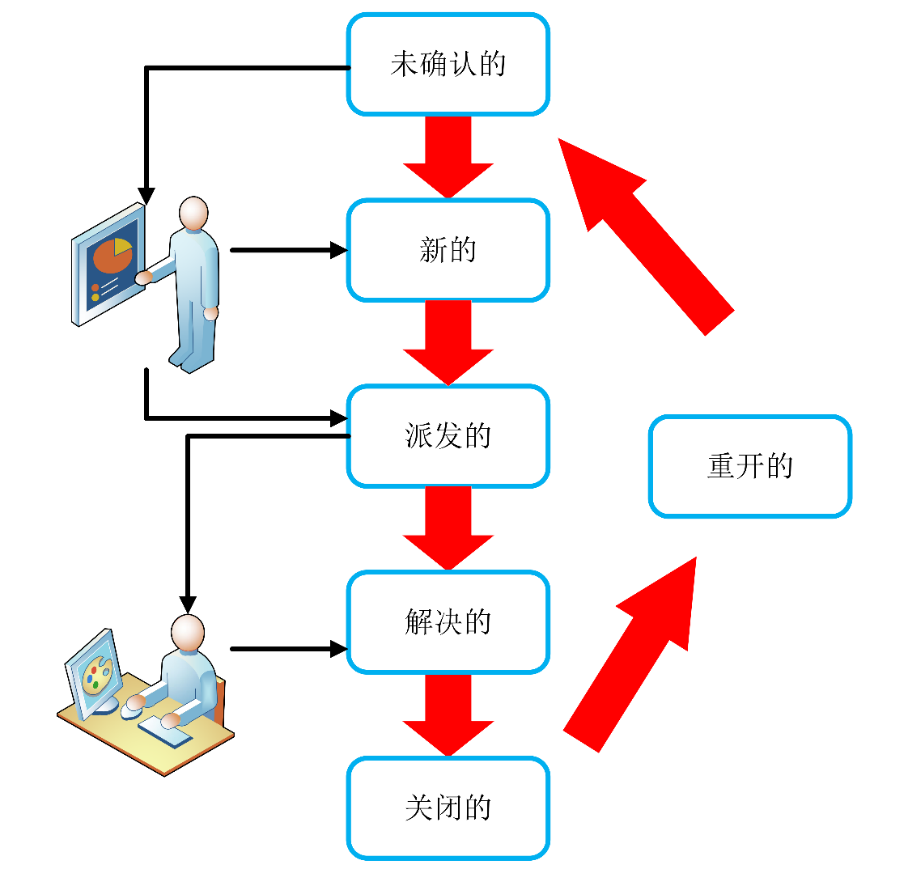
### 2.1.2 缺陷修复

缺陷报告在整个缺陷修复的过程中的状态是在变化着的[14]，图2.2展示了整个缺陷修复的流程。

当用户提交他撰写的缺陷报告后，缺陷报告的初始状态是未确认的（Unconfirmed）。随后会有派分者（Triager）确认这个缺陷报告描述的内容是否为缺陷，如果是，那么缺陷报告的状态变为新的（New）。然后也是由派分者负责把这个缺陷报告交给一个开发者进行修复。如果这个开发者完成了修复的内容，缺陷报告的状态又会变为已解决（Resolved），否则这个缺陷报告的状态会重新变为新的然后由派分者重新选择一位修复者。最终会由项目负责人判断这个缺陷有没有真的被解决，如果缺陷成功被修复，缺陷报告的状态最终会变为关闭的（Closed）。如果之后有人发现这个缺陷并没有被成功地修复，那么这个缺陷报告的状态则会变为重开（Reopened）然后重新进行上文描述的内容。

缺陷报告从提交到关闭的过程中主要包括三个阶段：缺陷理解、缺陷派分和缺陷修复。下面介绍这三个阶段：

（1）缺陷理解。派分者在确认缺陷之前需要仔细阅读缺陷报告的内容，他们需要提取出缺陷报告中重要的信息、找出并标注出缺陷报告的副本、给缺陷报告标注出优先级。这部分工作需要派分者具有丰富的经验和优秀的编码能力，但是由于很多开源项目缺乏

图2.2 缺陷修复的过程

这样的人员所以对于整个缺陷追踪系统造成了很大的问题。为了解决这些问题，研究人员提出过很多自动化算法，其中包括提取缺陷报告摘要、预测是否为缺陷、探测缺陷报告的副本、缺陷报告严重度的自动预测和缺陷报告优先级的自动生成。本文进行了其中的缺陷报告优先级自动生成的研究。本文提出的算法优于现存方法，希望能够为缺陷追踪系统作出贡献。

（2）缺陷派分。派分者在确认缺陷后需要将缺陷报告分配给最适当的修复者以确保缺陷可以得到及时的修复。但是如果派分者对修复人员不够了解或者对相应项目的理解不足，缺陷很容易被交付给一个不合适的修复人员。这样会导致修复者无法胜任这个任务而使得派分者需要重新分配，而这会导致浪费大量的时间，同时使得很多亟待修复的缺陷无法被解决。而且大量的缺陷报告也会使得派分者无法及时把它们分配给修复者。为了解决上述问题，研究人员也提出一些修复者推荐的算法。

（3）缺陷修复。当修复者收到派分者发送的缺陷报告后，他们就需要开始进行缺陷修复工作。修复者首先需要根据缺陷报告中的内容（主要是Description）找到缺陷所在的源程序文件，然后再分析代码找出其中存在的问题并修改错误代码，生成程序补丁。其中找到缺陷所在的程序是一件非常费力的工作，所以研究者提出很多缺陷自动定位算法。

## 2.2 文本分类算法

文本分类是自然语言处理领域的经典问题，文本分类算法被广泛应用于问答（Question Answering）、垃圾邮件检测（Spam Detection）、情感分析（Sentiment Analysis）和用户意图分类（User Intent Classification）等等。本文提出的算法的本质就是一个分类算法。所以本小节将介绍传统文本分类算法和基于深度学习（Deep Learning）的分类算法。

### 2.2.1 传统文本分类算法

传统机器学习算法的重点在于两个部分：特征提取和分类算法。

常见的机器学习特征提取方法是独热编码（One-hot Encoding）。这是一种较为简单的特征提取方法。如果数据中词表（Vocabulary）的大小为*N*，那么每个文本的特征长度就为*N*。当对某个文本进行编码的时候，*N*维特征向量中每个出现词位置上的值为1，否则为0。虽然独热编码可以很方便的得到文本的特征向量，但是这样的特征向量过于稀疏，会影响到算法的性能。

机器学习中常见的分类算法有：朴素贝叶斯算法、支持向量机算法（SVM）、K-近邻（KNN）算法、决策树和逻辑回归（Logistic Regression）等等。下面介绍前三种算法：

（1）朴素贝叶斯。这是一个简单的构建分类器的算法，它发源于古典数学理论而有着坚实的数学基础。虽然训练朴素贝叶斯的方法不是唯一的，但是朴素贝叶斯这个大家庭遵循着一个统一的原则：数据中所有的特征之间都是独立的。比如对于一个西瓜，它的花纹的宽度与瓜瓤的颜色间之间可能会有关系，但是如果用朴素贝叶斯对西瓜进行分类那么这两个特征则会被视为是无关联的。在实际应用中，朴素贝叶斯的参数估计法是极大似然法，换句话说，无需利用贝叶斯概率或者任何贝叶斯方法就可以使用朴素贝叶斯模型。尽管朴素贝叶斯分类器看起来设计得很简单甚至是非常的傻，但是它在很多复杂的情况下表现得仍然很好。这可能是因为在机器学习时代假设特征独立是最好的策略。朴素贝叶斯还有一个优点是只需要很少的数据就可以训练出分类所需要的参数。

（2）支持向量机。这是鲁棒性最强的二元分类算法之一，它的理论基础是VC理论。支持向量机把所有的训练集都映射到欧几里得空间中并使得两个类别间的距离达到最大。当预测新数据的时候，根据新数据处于空间里的哪一边来判断它的类别。除了进行线性分类，支持向量机还可以使用核技巧（Kernel Trick）实现非线性分类。在实际应用场景中，很多时候是没有标签的，所以支持向量机作为一种优秀的无监督分类算法被广泛地应用于工业界。

（3）K-近邻。这是一种无参数分类算法，可以同时被应用于分类和回归任务。它的独特之处在于对数据局部结构很敏感，这可能是其在向量空间模型中表现良好的原因。K-近邻算法的另一个优点是对于数据不平衡的鲁棒性。

### 2.2.2 深度文本分类算法

深度学习的本质在于自动寻找特征，所以基于深度学习的文本分类方法可以充分地捕捉数据中的信息。随着海量级别数据的出现以及硬件水平的快速提升，基于深度学习的文本分类方法获得了非常好的效果[15]。下面介绍其中常见的几类：

（1）基于前馈神经网络的模型。这是最简单的一种深度学习文本分类方法，但是仍然在很多数据集中获得了非常好的效果。这些模型首先利用词嵌入模型（比如：word2vec和Glove）获得每个词的特征向量，然后直接把某个文本中所有词的特征向量相加或者取平均获得所有文本的特征向量。随后把这些特征向量输入一层或者多层的前馈神经网络，最后在前馈神经网络的最后一层使用传统的机器学习分类器实现分类，比如逻辑回归、朴素贝叶斯和支持向量机。

（2）基于循环神经网络（RNN）的模型。循环神经网络擅长处理序列数据，所以这类模型把文本当作是一个词的序列并致力于从中挖掘出词之间的依赖和文本的结构信息。但是最简单的循环神经网络（Vanilla RNN）的表现不够理想，常常不如前馈神经网络。在循环神经网络的变种中，长短时神经记忆网络（LSTM）是最流行的一种，它更擅长于捕捉序列中的长距离依赖信息。长短时神经记忆网络引入记忆单元和三种门（输入门、输出门和遗忘门）解决了循环神经网络中的梯度消失和梯度爆炸问题。所以开发者利用它获取更多的信息从而提高了文本分类的精度。

（3）基于卷积神经网络的模型。循环神经网络擅长识别跨时间的模式而卷积神经网络精于捕捉跨空间的特征。而且卷积神经网络在捕捉局部和位置信息方面表现很好。而这些信息对于文本分类会很有帮助，所以卷积神经网络变成了文本分类领域非常流行的模型。Kalchbrenner等人[16]是第一个研究基于卷积神经网络的文本算法的团队，他们提出的算法名为DCNN。它的第一层是一个句特征矩阵，其中每个词对应着一个向量。随后的卷积框架使用动态最大池化层提取词之间短关系和长关系。动态最大池化层中的参数可以根据数据中句子的长度作出调整。

（4）基于胶囊神经网络（Capsule Neural Network）。虽然卷积神经网络中的池化操作可以识别明显的特征并降低计算复杂度，但是它会丢失有关空间关系的信息甚至会根据它们的方向和比例进行错误的分类。为了解决这个问题，胶囊网络应运而生。胶囊网络与普通神经网络的区别是其神经元是一个向量而不是一个标量，它的神经元叫做向量神经元（Vector Neuron）。近年来，有研究者利用胶囊网络对文本进行表征以实现文本分类算法。Zhao等人[17-19]提出了一个胶囊网络的变种进行文本分类。这个模型包含四层：n-gram卷积层、胶囊层、卷积胶囊层和全连接胶囊层。

（5）基于注意力机制的模型。注意力机制的灵感源自于人类对一个句子中不同词的注意力是不同的，它逐渐成为深度学习与自然语言处理领域的流行概念和有用的工具。语言模型中的注意力机制可以简要地解释为重要性权重的向量。当预测句子中的某个词的时候，可以根据这些权重向量来判断这个词与其他词的关联性。而研究者通过语言模型可以获得高质量的特征向量从而增强文本分类的效果。Shen等人[20]提出一个单向自注意力模型用于语言理解。其中各个部分之间的注意力是单向且多维度的。他们使用一个轻量级的神经网络用于训练句嵌入，这个网络完全基于注意力机制而不包含卷积神经网络和循环神经网络。

（6）基于记忆增强网络（Memory-Augmented Networks）的模型。注意力机制把模型的外部记忆保存在重要性权重向量而忽略了内部记忆。记忆增强网络改变网络结构解决了这个问题。Munkhdalai和Yu[21]提出一个用于文本分类和问答的记忆增强神经网络NSE（Neural Semantic Encoder）。NSE构建了可变大小的编码内存，该内存会随着时间的推移而变化和发展，并时刻保持对输入序列的理解。

（7）基于Transformer和预训练模型语言模型的模型。循环神经网络用于文本分类的一个瓶颈是文本的序列化处理的速度太慢。尽管卷积神经网络的序列性小于循环神经网络，但是随着句子长度的增加，它捕捉词语间关系的计算量也会很大而导致速度过慢。Transformer使用自注意力机制并行地捕捉文本间的信息解决了这个问题。Transformer也就使得大型数据可以在GPU上高效地训练。2018年是大型基于Transformer的预训练语言模型（PLMs）的流行元年。与之前基于卷积神经网络和长短时记忆网络的语言模型相比，基于Transformer的PLMs有着更深的模型并利用预测函数在非常巨大的无标签数据中进行训练。这些PLMs可以利用简单的操作进行微调（Fine-tune）而在一些下游任务中包括文本分类获得最前沿水平（State-of-the-art）。也就是说，PLMs是无监督的但是微调后是有监督的。PLMs可以被分为两类，自动回归（Autoregressive）和自动编码（Autoencoding）。GPT（Generative Pre-Training）属于早期的自动回归PLMs，它使用12层Transformer模块和单向预测函数（根据左边的内容预测右边的词）训练模型。BERT[22]（Bidirectional Encoder Representations from Transformers）是最常用的自动编码PLMs。BERT使用掩码语言模型（Masked Language Model）任务进行训练，掩码语言模型随机遮掩句子中的词语然后把这些词作为模型需要预测的内容。与GPT不同的是，BERT使用双向Transformer作为模型基础。

## 2.3 图神经网络

近年来，图神经网络成为人工智能领域的热门研究方向，且在很多实际应用领域大放异彩[23]。所以本节将介绍图神经网络。

### 2.3.1 图神经网络的背景和用途

神经网络的成功应用推动了模式识别和数据挖掘的研究，比如图像分类、语音识别和自然语言理解等等。深度学习的成功归结于快速发展的计算资源和其自身从数据（比如图片、文本和视频）提取关键信息的能力，但是这些数据都是属于欧几里得空间的，而有的数据是无法在欧几里得空间生成的。具有复杂关系和依赖的数据只能用图进行表征。不幸的是传统机器学习方法很难处理复杂的图数据。为了能够更好地利用图数据，很多研究者尝试利用深度学习处理非欧几里得空间里的数据。

图神经网络有着非常广阔的应用空间，下面介绍其中的几种应用：

（1）推荐系统领域。推荐系统的目的是判断用户是否喜欢某个商品。所以可以首先根据用户与用户、用户与商品和商品与商品之间的关系构建出一张图，然后把推荐问题转化为预测用户与商品间链接的问题。有学者结合图卷积神经网络和循环神经网络学习用户对商品的评分。

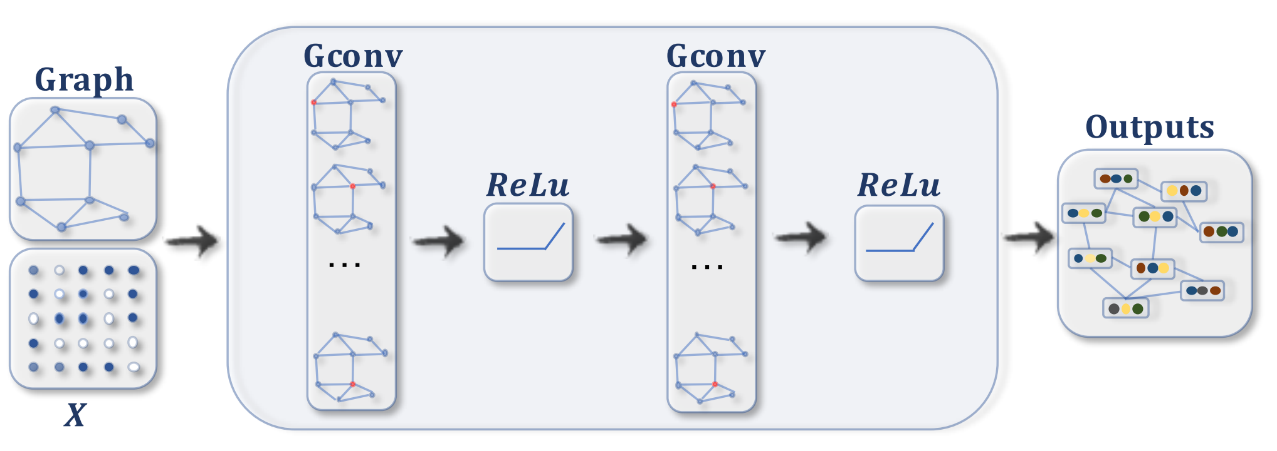
（2）交通领域。随着城市的扩张，旧时规划的街道难以支撑如今庞大的车流量。所以准确预测交通网络中的交通速度、道路密度和交通流量成为一个重要的研究课题。有研究者提出用基于图的时空神经网络方法解决这个问题。他们把一张时空图作为模型的输入，时空图中的节点是道路中的传感器，每个结点都包含着时间序列的特征。模型的目标是预测某条道路在某段时间内的平均车速。

（3）化学领域。化合物是由原子和化学键构成的，所以它们本质上就应该是图数据结构的。药物的有效成分就是某种化合物，但是枚举法寻找可以作为药物的化合物是一件非常低效的事。所以有研究者通过图神经网络捕捉已存在的药物化合物中的特征，然后预测有可能成为新药物的化合物的结构。除此之外，图神经网络还可以进行化合物筛选、蛋白质相互作用点检测、疾病预测等等。

### 2.3.2 图神经网络的分类

图神经网络可以大致被分为四类，循环图神经网络（RecGNNs）、卷积图神经网络（ConvGNNs）、图自动编码（GAEs）和时空图神经网络（STGNNs）：

（1）RecGNNs主要是图神经网络的初始研究内容。RecGNNs的设计目的是使用循环神经网络学习结点表征。研究者们假设图中的一个节点不断与其邻居交换信息/消息，直到达到稳定的平衡为止。RecGNNs中信息传递的思想启发了后来的研究，随后ConvGNNs被设计出来。

（2）ConvGNNs将卷积操作从网格化的数据带到图数据。ConvGNNs的主要思想是整合某个结点和其邻居结点的特征获得结点的表征。ConvGNNs是构建很多其他复杂图神经网络的重要组成部分。图2.3展示了一个结点分类模型。图2.4展示了一个图分类的模型。

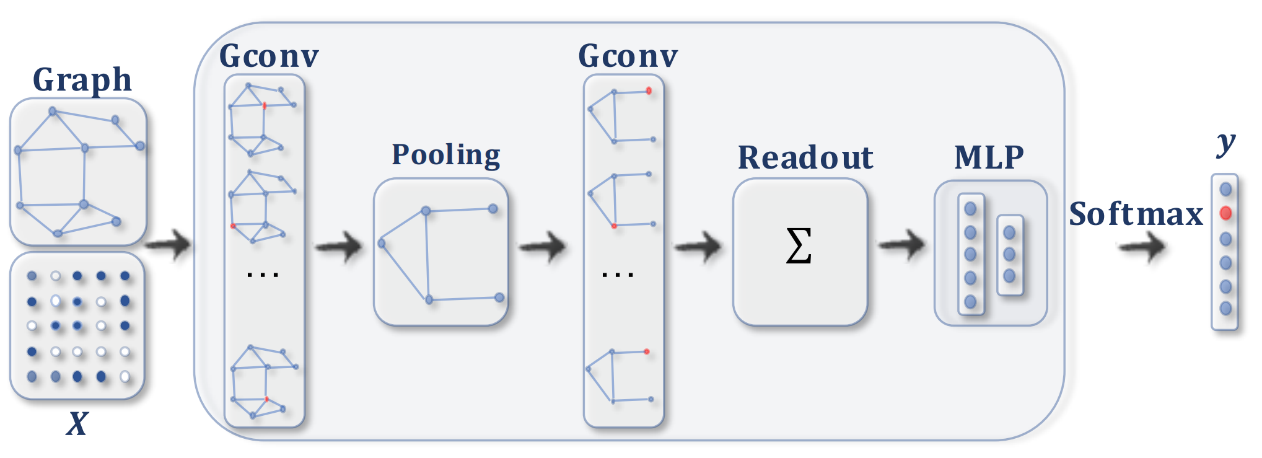
图2.3 ConvGNNs用于结点分类[23]

图2.4 ConvGNNs用于图分类[23]

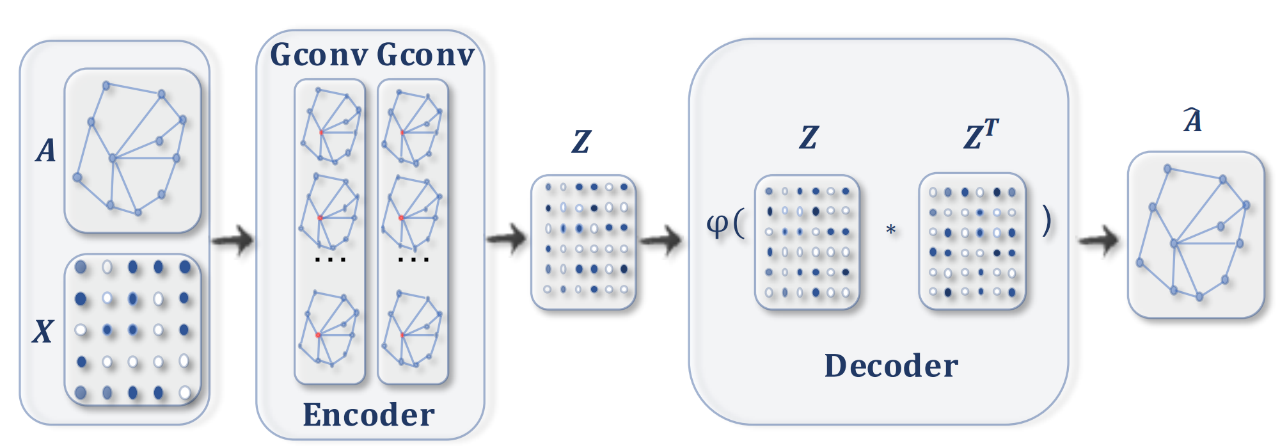
（3）GAEs是无监督学习框架，它们把结点或者图编码入一个隐藏向量空间然后根据编码信息重构出图数据。所以GAEs被用于学习网络嵌入和图生成分布。关于网络嵌入，GAEs通过重构如邻接矩阵这样的信息学习结点的表征。关于图生成，有些方法一步步生成图的结点和边，有的直接生成最终的图。图2.5展示了一个网络嵌入模型。

图2.5 网络嵌入模型[23]

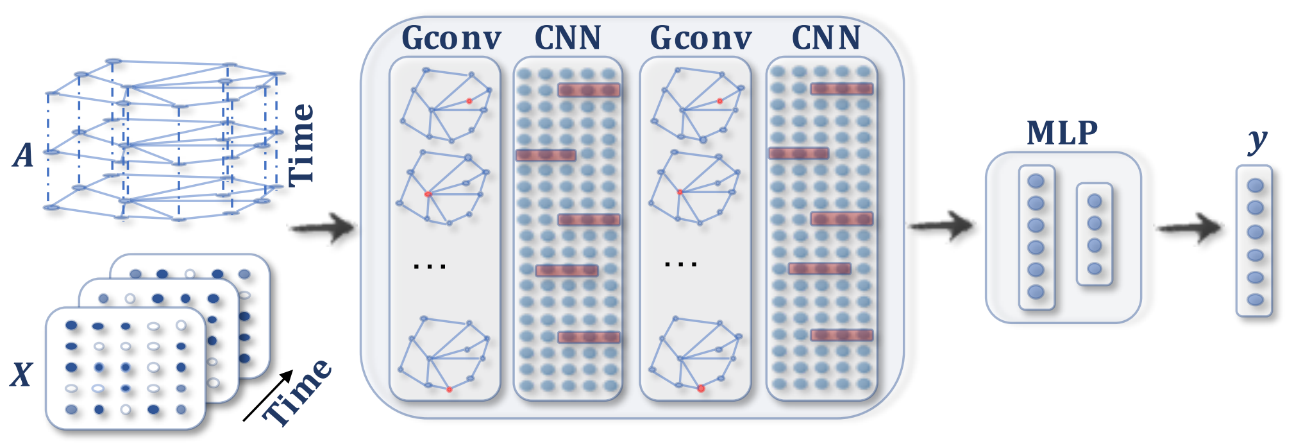
（4）STGNNs被用于从时空图中提取隐藏模式。这使得STGNNs在某些任务上大放异彩，比如交通速度预测[24]、驾驶员操作预判[25]和人类动作识别[26]等等。STGNNs的成功之处在于同时考虑时间和空间依赖。图2.6展示了一个图预测模型。

图2.6 图预测模型[23]

### 2.3.3 图神经网络的未来工作

深度学习的成功在于深层网络的设置，但是现存的经验和作者的实验结果表明图卷积神经网络的性能随着网络层数的增加而急剧下降。从理论上分析，随着卷积层数的增加，相邻结点的表征会越来越相似直至最后收敛于某个点。所以图神经网络的研究者未来需要研究图神经网络是否需要深模型。

以往的神经网络分类模型是需要大量的标注数据的，而这大大限制了模型的实际应用，因为找到世界上所有的标注数据是不现实的。图神经网络擅长于使用少量数据达到令人满意的效果，所以它在未来会被应用于很多领域。

## 2.4 本章小结

本章介绍了论文相关概念以及技术。作者首先介绍了缺陷追踪系统，本文的研究课题就是为了实现其中缺陷报告优先级标注步骤的自动化。然后介绍了文本分类算法，因为本文提出的算法的本质是一个分类算法。最后，算法是基于图卷积神经网络的，所以最后介绍了图神经网络。

# 第3章 基于图卷积神经网络的缺陷报告优先级自动生成

本文提出基于图卷积神经网络的缺陷报告优先级自动生成算法（Priority Prediction algorithm by using Weighted Graph Convolutional Networks，以下简称为PPWGCN），本章将详细介绍PPWGCN。为了使每个步骤更加具体化而更容易理解，本章将把图2.1中的缺陷报告作为解释性的例子贯穿全文。

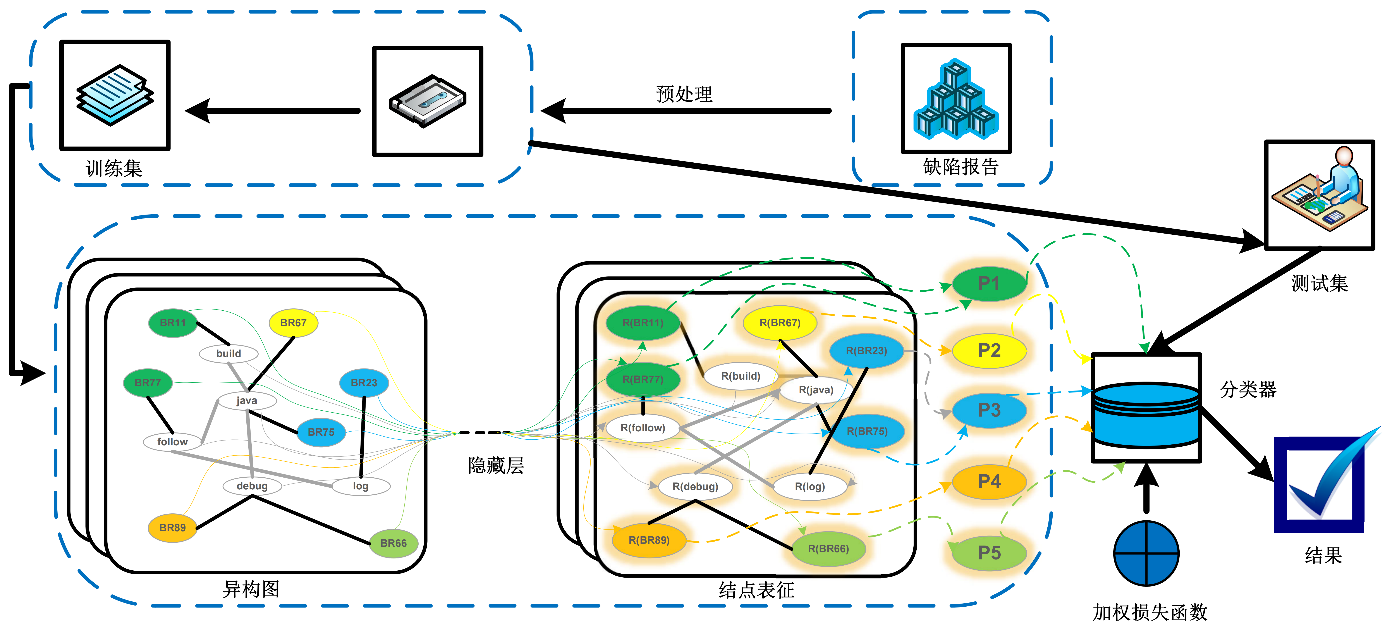
## 3.1 PPWGCN算法概述

为了提高自动化优先级标注的效率，作者提出了一个新的算法PPWGCN。图3.1展示了PPWGCN的整个流程。其中以BR开头的结点是文本结点，其他的是词结点。灰色的直线代表词与词间的边，黑色的直线代表词与文本间的边。R（x）代表x的词嵌入或者是文本嵌入（这取决于x是单词或者是文本）。各种颜色的曲线表示图卷积神经网络中信息的流动。首先简要介绍每个步骤：

（1）把缺陷报告的五个部分Severity、Product、Component、Description和Summary作为一个整文本，然后使用去除停用词和去除稀有单词（Rare Word Removal）等方法对数据进行预处理。因为预处理可以显著地减少模型需要的内存空间，并提高算法结果的精度。

（2）构建了一张拥有词结点和文本结点两种结点的异构图。该图有两种带边权的边，词与词之间的边和词与文本之间的边，文本与文本之间没有边。其中词与词之间的边权由点间互信息（Pointwise Mutual Information）计算得出，词与文本间的边权由TF-IDF（Term Frequency–Inverse Document Frequency）得出。其中的词指对缺陷报告进行预处理后得到的词，文本代指实验中用到的缺陷报告，每个文本结点都包含着一个优先级标签。

（3）把这张异构图输入一个两层图卷积神经网络中并使用加权交叉熵损失函数训练出一个分类器。当完成训练过程后，最终使用这个分类器来预测测试集中缺陷报告的优先级。因此作者把缺陷报告优先级自动生成问题转化为图结点分类问题。虽然每个缺陷报告之间没有直接的联系，但是经过两层图卷积神经网络的传播之后任意两个缺陷报告间可以实现信息的交互。不仅如此，图卷积神经网络的特性是训练速度快和在低训练集比例下也能表现得令人满意等，而软件工程领域缺少数据，所以这个神经网络特别适合用来解决缺陷报告优先级自动生成问题。

图3.1 PPWGCN流程图

## 3.2 数据预处理

本文的数据是来自四个开源项目Mozilla、Eclipse、GCC和Netbeans的缺陷报告。作者提取出缺陷报告中的五个部分Severity、Product、Component、Description和Summary作为数据的内容部分，然后将缺陷报告中的优先级作为数据的标签部分。本节将介绍如何对数据的内容进行预处理。作者将以图2.1中展示的缺陷报告作为描述实例。

首先把缺陷报告中的五个部分摘录下来合成一个文本，由此便可以得到原始的文本数据：*button, does, not, support, wrap, style, while, label, supports, wrap, style, request, wrap, style, for, button, enhancement, swt, platform*。这个缺陷报告描述了关于按钮使用的问题，所以它的优先级不算很高，为P4。

得到文本后把文本切分成tokens，使得文本在空格字符除被拆分。随后把所有的英文字符转化为小写。计算机是无法直接处理自然语言的，所以需要将自然语言转化为01代码。为了减少内存的使用，一般的做法是将文本切分为tokens后用数字为它们编码。对于中文自然语言处理，tokens为词序列，每个token可能包含两个字也有可能包含三个字。而本文用到的缺陷报告数据是英文的，对于英文是不需要分词的，直接将每个单词隔开即可。

然后去除原始文本中的停用词（比如not、are和for等等），因为停用词对算法性能的提升是没有帮助的，它们广泛地存在于所有数据中而无法使相应的数据具有标识性的特征。作者使用NLTK（Natural Language Toolkit）Python库对停用词进行过滤。NLTK中包含的停用词有：‘ourselves’, ‘hers’, ‘between’, ‘yourself’, ‘but’, ‘again’, ‘there’, ‘about’, ‘once’, ‘during’, ‘out’, ‘very’, ‘having’, ‘with’, ‘they’, ‘own’, ‘an’, ‘be’, ‘some’, ‘for’, ‘do’, ‘its’, ‘yours’, ‘such’, ‘into’, ‘of’, ‘most’, ‘itself’, ‘other’, ‘off’, ‘is’, ‘s’, ‘am’, ‘or’, ‘who’, ‘as’, ‘from’, ‘him’, ‘each’, ‘the’, ‘themselves’,‘until’等等。

作者同时也去除文本中的稀有单词。稀有单词会减少自然语言处理领域模型的精度和效率，因为它们的数量非常稀少而无法作为有效的特征。而且很多稀有单词是用户输入的拼错的单词或者是乱打的单词，它们会引导模型往失败的方向走去，所以去掉了出现次数少于五的单词。

最终上述例子中的原始文本变为：*button, support, wrap, style, label, supports, wrap, style, request, wrap, style, button, enhancement, swt, platform*。其中*does、not、while、for*作为停用词被去除。

## 3.3 文本异构图构建

在处理完数据后首先构建出一个文本异构图，其中代表了图中的结点，代表了图中的边。PPWGCN中的异构图中有两类结点：词结点和文本结点。所有结点的个数由两个部分组成，文本的个数（即缺陷报告的总数）和词表（所有特殊词的个数）中单词的个数。其中文本结点中的训练集部分是包含一个优先级标签的，测试集部分没有标签。

在异构图中一共有两类边，词与词之间的边和词与文本之间的边。文本和文本之间是没有边的。对于词与词之间的边，使用滑动窗口法去捕捉每个文本中的共现词信息。其中的滑动窗口表示规定一个窗口包含*n*个词然后从文本的前*n*个词以一个词为单位不断地向后滑动，每滑动一次会产生一个新的滑动窗口同时分析每个滑动窗口中的词。点间互信息是用来衡量词与词之间关系非常经典而有效的方法，它被广泛地应用于传统自然语言处理领域。而词共现信息可以捕捉文本中的全局特征从而帮助PPWGCN获得更多有效的信息。所以使用点间互信息计算词与词之间的边权。两个词之间的点间互信息的计算方式如下：

（3-1）

（3-2）

（3-3）

其中代表同时含有单词和的滑动窗口的总个数，表示所有数据中的滑动窗口的个数,表示所有的数据中包含单词的滑动窗口的数量。

由式（3-1）可知点间互信息的值是可能为负数的，负值表示词与词间的语义联系是弱的而正值代表着文本中词与词间的语义联系是较强的。作者构建异构图的目的是提取出缺陷报告间的关联信息，而负值会产生噪声的。所以为了提取出词与词之间的强联系关系，只将正点间互信息值作为边权。

作者使用TF-IDF算法获取词与文本间的边权。TF-IDF是信息检索领域的算法，它的各种变种被用于搜索引擎中以获得用户查询的文本与数据库中文件的相关性。

TF-IDF是一种用于判断某个词关于一个文集中某个文本的重要性的算法，它本质上是基于词频的。算法的核心思想是在某文本中出现次数越多的词的重要性越大，但是也会与在文集中出现的频率成反比。因此定义词与文本的边权：

（3-4）

其中代表文本也就是数据中的缺陷报告，表示数据中的某个词，代表单词出现在文本中的次数。

IDF是对文本总数除以包含单词的文本数取对数的结果，IDF越大这个单词的重要性就越大。

最终可以根据上文归纳出异构图的边权计算方式如式（3-5）。

（3-5）

## 3.4 基于图卷积神经网络的文本分类

在构建好文本异构图后，把这张图输入一个由简单的两层图卷积神经网络构成的分类器中：

（3-6）

其中是规范化对称邻接矩阵（Symmetric Normalized Laplacian）。它是图卷积神经网络中最重要的部分，它可以表示两个结点间的关系并使得图中的邻居信息流动。

图神经网络可以被分为两大类别，顶点域（Spatial Domain）方法和频域（Spectral Domain）方法，本文用到的图卷积神经网络属于频域方法。其本质是依赖图谱领域的理论实现对图数据的卷积操作。

图数据中的傅里叶变换（Fourier Transformation）最早是由研究GSP（Graph Signal Processing）的研究者提出并给出推导证明的。而实现谱域的卷积操作，最重要的是得到可以被特征分解的矩阵。规范化对称邻接矩阵属于拉普拉斯矩阵，是一个半正定矩阵，有n个线性无关的特征向量且所有的特征值都是非负的，所以它被应用于谱域图神经网络。

在实际运用中，拉普拉斯矩阵的特征向量被作为傅里叶变换的基，特征值被当作频率。这样做的抽象理解是：谱域是多个简单元素的线性组合，若它们是正交的即可被作为基。

是一个邻接矩阵，为了捕捉自信息（Self-information），中的主对角线被设置为1。是一个度矩阵，它由得出。是算法的结果，其中n是结点的个数，也就是文本的总数加上词表的个数；表示类别标签的总和（五个严重度标签：P1、P2、P3、P4和P5）。

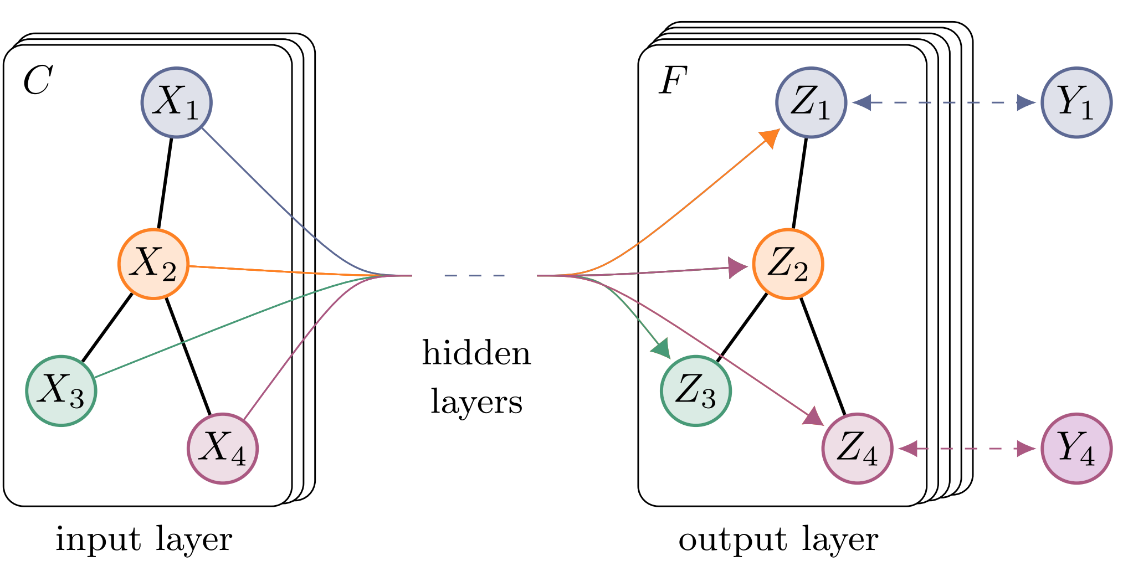
与单层图卷积神经网络不同的是，双层图卷积神经网络可以使得信息在结点间播，它们最大的间隔可以为2。所以信息可以在文本结点中传播，尽管文本异构图中文本与文本结点之间没有直接连接。图3.2展示了一个双层图卷积神经网络。

图3.2 双层图卷积神经网络[27]

PPWGCN第一层的定义如下：

（3-7）

其中的是一个损失函数，中文翻译为线性整流函数（Rectified Linear Unit）。线性整流函数相比于其他的损失函数比较简单，它的输出为输入与0的最大值。线性整流函数有以下三点优点：

* 因为没有诸如指数函数这种复杂的计算和自身的分散性，所以线性整流函数的计算成本比较低。
* 避免了出现梯度消失和梯度爆炸的问题。
* 线性整流函数的设计符合仿生学原理，因为生物神经元的信息编码一般是比较稀疏的，大脑在同一时刻只有不到5%的神经元处于活跃状态。所以它可以胜任人工神经网络中的激活函数。

线性整流函数的定义如下：

（3-8）

式（3-7）中的是一个特征矩阵，它是一个单位矩阵。所以在模型训练的过程中，这个特征矩阵是不增加计算量的。作者也考虑过用一些词嵌入模型代替单位矩阵，因为单纯的单位矩阵无法表征词或者文本结点的信息，而一些流行的词嵌入模型比如word2vec可以在训练PPWGCN之前就获得结点的语义信息（包括词结点和文本结点）。所以作者尝试使用论文中四个开源项目的缺陷报告数据训练出300维度的word2vec模型，异构图中的词结点直接由词嵌入表示，而文本结点取其中所有词的词嵌入的平均值。很不幸的是这样的做法并不理想，换成词嵌入特征矩阵后增大了很多的计算量而使得模型训练的速度变得很慢，而且最重要的是模型的结果并没有因为增加了词嵌入信息而变好。作者觉得这可能是因为图卷积神经网络通过拉普拉斯矩阵和卷积操作已经使得图中的信息得到充分的传播。加入前馈神经网络后，图中的结点的特征已经被非常完备地提取，不同类别的结点可以被轻易分辨出来，而加上词嵌入信息后无疑增加模型的负担，起到了画蛇添足的作用。

式（3-7）中的是一个权重矩阵，其中的代表隐藏层的大小，设置为200。第二层的公式如下：

（3-9）

其中的是第二个权重矩阵。在得到最终的结点表征后利用softmax函数实现分类：

（3-10）

（3-11）

其中的代表五个类别标签P1、P2、P3、P4和P5中的一个，为了解决数据不平衡的问题，作者在损失函数中引入了标签罚项。在下一章可以看出优先级预测的数据是非常不平衡的，这会导致数量多的标签的效果非常好而数量小的标签的效果不好，前面的研究者曾尝试解决这个问题但是效果并不能令人满意。所以引入标签罚项是PPWGCN的一个重要的创新点。损失函数定义如下：

（3-12）

这个损失函数本质上是一个交叉熵损失函数。

其中表示训练集，代表标签的数量也就是5，表示一个标签指示矩阵，代表每个标签的权重。权重矩阵的定义如下：

（3-13）

其中表示标签的个数。

尽管文本与文本间没有直接的边，但是信息传播使得任意两个文本结点间有了联系，这可能也是简单的PPWGCN表现出色的原因。所以相信PPWGCN是一个用于自动生成缺陷报告优先级的优秀算法。

## 3.5 本章小结

本章详细介绍了PPWGCN。首先是数据预处理，对缺陷报告的五个部分Severity、Product、Component、Description和Summary进行去停用词和去稀有词处理。然后是构建文本异构图，其中文本结点与文本结点间没有边，词结点与词结点之间的边权由点间互信息计算得出，词结点与文本结点间的边权由TF-IDF算法得出。最终将图输入一个两层的图卷积神经网络中实现缺陷报告优先级的自动生成，并引入加权损失函数。

# 第4章 实验设计与分析

为了验证PPWGCN的效果，本文爬取四个开源项目Mozilla、Eclipse、GCC和Netbeans中的缺陷报告为实验数据，并设计出三个实验以更好地评估算法的效果。三个实验分别为对比PPWGCN与两个基线算法的效果、评估PPWGCN在低训练集比例下的效果和验证加权损失函数能否解决数据不平衡问题。作者从动机、实验步骤和结果分析三个方面描述实验。

## 4.1 实验设置

为了详细地描述本文实验的内容，本节将介绍实验用到的数据、评价指标和参数设置。

### 4.1.1 评价指标

为了评估PPWGCN的效果并分析实验的结果，作者选取F-measure值作为评估指标。精确率（P），召回率（R）和F-measure定义如下：

（4-1）

（4-2）

（4-3）

其中代表某个类别标签，（True Positive）代表被预测为标签且预测正确的数量；（False Positive）表示被预测为标签但是预测错误的数量；（False Negative）代表被预测为但是预测错误的数量。

精确率（P）衡量的是模型的查准率而召回率（R）衡量的是模型的查全率，但是单纯重视他们中的某一个是有失偏颇的。极端情况下，如果对某标签只预测出其一个实例，那么这个标签的精确率就会很高而召回率很低；相反如果把所有的实例都预测为某标签，那么这个标签的召回率会很高但是精确率很低。所以最终选择了综合考虑精确率和召回率的F-measure值作为评价指标。

### 4.1.2 实验数据

为了验证算法的有效性，作者在四个开源项目的缺陷报告中进行了一系列的实验。收集了从2000年2月到2020年9月的数据，并把具有相同标签的缺陷报告归为一类。

表4.1和表4.2具体展示了数据，由表4.1可以看出数据是非常不平衡的。以Eclipse为例，P3标签的缺陷报告的数量为90026而P4标签的数量仅为2828。这会给机器学习模型带来灾难性的影响，因为模型总会朝着数量多的标签学习。PPWGCN引入标签罚项解决了这个问题，本章将会给出实验验证。

表4.2展示了不同项目文本的词表大小以及文本的平均长度。由表4.2可以看出四个开源项目中Mozilla的词表最大，为68314；GCC的词表最小，为38096。Eclipse的平均文本长度最短，为168.82；Netbeans的平均文本长度最长，为356.03。

表4.1 项目中不同标签的缺陷报告数量

|  |  |  |
| --- | --- | --- |
| 项目名称 | 优先级标签 | 缺陷报告的数量 |
| Mozilla | P1 | 30247 |
| P2 | 9818 |
| P3 | 13000 |
| P4 | 4974 |
| P5 | 3771 |
| 总计 | | 61810 |
| Eclipse | P1 | 4850 |
| P2 | 9964 |
| P3 | 90026 |
| P4 | 2828 |
| P5 | 3270 |
| 总计 | | 110938 |
| Netbeans | P1 | 11946 |
| P2 | 18934 |
| P3 | 18399 |
| P4 | 2873 |
| P5 | 0 |
| 总计 | | 52152 |
| GCC | P1 | 3905 |
| P2 | 12418 |
| P3 | 16164 |
| P4 | 384 |
| P5 | 241 |
| 总计 | | 33112 |

和基线算法DRONE一样，随机选取50%的数据作为训练集，其余的50%数据作为测试集。

表4.2 数据的统计结果

|  |  |  |
| --- | --- | --- |
| 项目名称 | 词表的大小 | 文本的平均长度 |
| Mozilla | 68314 | 177.59 |
| Eclipse | 56742 | 168.82 |
| Netbeans | 47928 | 356.03 |
| GCC | 38096 | 247.05 |

### 4.1.3 参数设置

对于PPWGCN中的图卷积神经网络的参数，作者设置隐藏层的维度为200，滑动窗口的大小为20。设置模型训练的学习率（Learning Rate）为0.02，丢失率（Dropout Rate）为0.5。

现存研究表明图卷积神经网络训练的速度相对比较快，训练的轮回次数不需要很多。实验表明训练轮回（Epoch）在200后的增加并不能带来模型效果的提升，故本文的实验以200为模型的最大轮回数。作者在实验过程中发现轮回数达到一百时就接近最优结果了。

## 4.2 PPWGCN与两个基线算法的效果的对比

### 4.2.1 动机与实验步骤

PPWGCN利用图卷积神经网络实现了高效的优先级自动生成，并融入加权损失函数以应对数据不平衡的问题。且PPWGCN构建了一张文本异构图，故其可以充分提取数据中的信息包括全局信息。

为了验证它，作者选取了两个领域内最新最优秀的方法DRONE[1]和cPur[3]作为基线算法。DRONE出自Empirical Software Engineering，软件工程领域国际顶级期刊；cPur出自IEEE Transactions on Reliability，安全可靠性方向的国际顶级期刊。

作者复现了DRONE[1]和cPur[3]，DRONE利用线性回归并训练出四个阈值以实现缺陷报告优先级的自动生成，cPur结合情感分析使用卷积神经网络实现缺陷报告优先级的自动生成，然后在四个开源项目上运行三个算法。最终以F-measure值为评估指标对实验结果进行分析。

### 4.2.2 结果与分析

表4.3，表4.4和表4.5分别展示了PPWGCN和两个基线算法的结果，可以明显地看出PPWGCN优于其余两个算法。详细分析如下：

表4.3 PPWGCN的实验结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 项目名称 | 优先级标签 | 精确率(%) | 召回率(%) | F-measure值(%) |
| Mozilla | P1 | 76.22 | 59.15 | 66.59 |
| P2 | 30.82 | 37.75 | 33.84 |
| P3 | 56.19 | 64.85 | 60.17 |
| P4 | 37.00 | 53.26 | 43.62 |
| P5 | 61.27 | 66.01 | 63.52 |
| 平均值 | | 52.30 | 56.20 | **53.55** |
| Eclipse | P1 | 16.70 | 34.82 | 22.49 |
| P2 | 19.34 | 32.06 | 24.08 |
| P3 | 88.89 | 67.33 | 76.58 |
| P4 | 10.85 | 29.94 | 15.87 |
| P5 | 15.10 | 37.59 | 21.47 |
| 平均值 | | 30.18 | 40.35 | **32.10** |
| Netbeans | P1 | 47.98 | 50.23 | 49.02 |
| P2 | 51.12 | 46.67 | 48.73 |
| P3 | 62.52 | 54.86 | 58.40 |
| P4 | 18.79 | 40.21 | 25.55 |
| P5 | 0 | 0 | 0 |
| 平均值 | | 36.08 | 38.39 | **36.34** |
| GCC | P1 | 31.31 | 43.62 | 36.41 |
| P2 | 61.88 | 57.79 | 59.73 |
| P3 | 68.99 | 61.30 | 64.87 |
| P4 | 10.65 | 29.13 | 15.51 |
| P5 | 6.72 | 17.41 | 9.64 |
| 平均值 | | 35.91 | 41.85 | **37.23** |

（1）对于Mozilla项目，算法对于P1、P2、P3、P4和P5标签的预测F-measure值分别为66.59%、33.84%、60.17%、43.62%和63.52%，平均F-measure值为53.55%。其中P1的结果最好，P2的结果最差。

在PPWGCN与DRONE的对比中，作者发现DRONE在P1标签中的表现比本文提出的算法好，这是数据不平衡带来的负面影响。PPWGCN的平均F-measure值比DRONE高28.60%。

表4.4 DRONE的实验结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 项目名称 | 优先级标签 | 精确率(%) | 召回率(%) | F-measure值(%) |
| Mozilla | P1 | 100 | 51.12 | 67.65 |
| P2 | 21.77 | 25.66 | 23.56 |
| P3 | 16.99 | 31.47 | 22.06 |
| P4 | 5.38 | 34.30 | 9.31 |
| P5 | 100 | 1.11 | 2.19 |
| 平均值 | | 48.83 | 28.73 | 24.95 |
| Eclipse | P1 | 0 | 0 | 0 |
| P2 | 0.55 | 1.90 | 0.86 |
| P3 | 81.45 | 45.88 | 58.70 |
| P4 | 0.34 | 0.61 | 0.44 |
| P5 | 0.03 | 3.51 | 0.07 |
| 平均值 | | 16.48 | 10.38 | 12.01 |
| Netbeans | P1 | 42.59 | 70.21 | 53.02 |
| P2 | 95.46 | 72.35 | 82.32 |
| P3 | 0 | 0 | 0 |
| P4 | 0 | 0 | 0 |
| P5 | 0 | 0 | 0 |
| 平均值 | | 27.61 | 28.51 | 27.07 |
| GCC | P1 | 59.49 | 98.85 | 74.28 |
| P2 | 0 | 0 | 0 |
| P3 | 95.49 | 96.12 | 95.81 |
| P4 | 0 | 0 | 0 |
| P5 | 0 | 0 | 0 |
| 平均值 | | 31.00 | 38.99 | 34.02 |

在PPWGCN与cPur的对比中，作者发现cPur只能预测出P1标签 。原因是不平衡的数据对神经网络的学习造成了巨大的影响，模型随着损失函数值的降低朝着数量多的标签的方向前进。PPWGCN不仅在平均F-measure值上超过cPur40.41%而且可以更加均衡地预测所有标签。因为PPWGCN可以充分地捕捉文本与词、词与词和文本与文本之间的关系。

所以PPWGCN在Mozilla项目中的结果超过DRONE和cPur算法，且结果相对更加平衡。

（2）对于Eclipse项目，算法对于P1、P2、P3、P4和P5标签的预测F-measure值分别为22.49%、24.08%、76.58%、15.87%和21.47%，平均F-measure值为32.10%。其中P3的结果最好，P4的结果最差。

表4.5 cPur的实验结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 项目名称 | 优先级标签 | 精确率(%) | 召回率(%) | F-measure值(%) |
| Mozilla | P1 | 48.94 | 100 | 65.71 |
| P2 | 0 | 0 | 0 |
| P3 | 0 | 0 | 0 |
| P4 | 0 | 0 | 0 |
| P5 | 0 | 0 | 0 |
| 平均值 | | 9.79 | 20.00 | 13.14 |
| Eclipse | P1 | 0 | 0 | 0 |
| P2 | 0 | 0 | 0 |
| P3 | 81.15 | 100 | 89.59 |
| P4 | 0 | 0 | 0 |
| P5 | 0 | 0 | 0 |
| 平均值 | | 16.23 | 20.00 | 17.92 |
| Netbeans | P1 | 0 | 0 | 0 |
| P2 | 40.54 | 80.61 | 53.95 |
| P3 | 59.23 | 46.71 | 52.23 |
| P4 | 0 | 0 | 0 |
| P5 | 0 | 0 | 0 |
| 平均值 | | 19.95 | 25.46 | 21.24 |
| GCC | P1 | 0 | 0 | 0 |
| P2 | 0 | 0 | 0 |
| P3 | 48.81 | 100 | 65.60 |
| P4 | 0 | 0 | 0 |
| P5 | 0 | 0 | 0 |
| 平均值 | | 9.76 | 20.00 | 13.12 |

在PPWGCN与DRONE的对比中，作者发现PPWGCN可以预测出所有的标签但是DRONE 只能预测P2、P2、P4和P5标签。PPWGCN在平均F-measure值指标上也比DRONE高20.09%。

在PPWGCN与cPur的对比中，作者发现cPur只能预测出P3标签。由此可见cPur在面对不平衡数据时几乎毫无招架之力。PPWGCN在平均F-measure值指标上比cPur高14.18%。

所以PPWGCN在Eclipse项目中的结果超过DRONE和cPur算法，且结果相对更加平衡。

（3）对于Netbeans项目，算法对于P1、P2、P3、P4和P5标签的预测F-measure值分别为49.02%、48.73%、58.40%、25.55%和0%，平均F-measure值为36.34%。其中P3的结果最好，P5的结果最差。

在PPWGCN与DRONE的对比中，作者发现PPWGCN可以预测出除了P5的所有标签但是DRONE 只能预测P1和P2标签。PPWGCN在平均F-measure值指标上也比DRONE高9.27%。

在PPWGCN与cPur的对比中，作者发现cPur只能预测出P2和P3标签。PPWGCN在平均F-measure值指标上比cPur高15.10%。

所以PPWGCN在Netbeans项目中的结果超过DRONE和cPur算法，且结果相对更加平衡。

（4）对于GCC项目，算法对于P1、P2、P3、P4和P5标签的预测F-measure值分别为36.41%、59.73%、64.87%、15.51%和9.64%，平均F-measure值为37.23%。其中P3的结果最好，P5的结果最差。

在PPWGCN与DRONE的对比中，作者发现PPWGCN可以预测出所有的标签，但是DRONE只能预测P1和P3标签。PPWGCN在平均F-measure值指标上也比DRONE高3.21%。

在PPWGCN与cPur的对比中，作者发现cPur只能预测出P3标签。PPWGCN在平均F-measure值指标上比cPur高24.11%。由此可见数据不平衡的负面作用是非常大的，因而需要研究者应对这个问题。

所以PPWGCN在GCC项目中的结果超过DRONE和cPur算法，且结果相对更加平衡。

与基线算法相比，本文提出的算法考虑了缺陷报告中的五个部分并通过构建一个文本异构图捕捉全局共现信息。同时图卷积神经网络可以通过拉普拉斯平滑操作使得同一个标签结点的特征变得明显然后进行分类，这种操作混合了某节点及其邻居结点的特征。而且使用加权损失函数应对不平衡数据。所以PPWGCN获得了较为平衡且显著超过两个基线算法的结果。

## 4.3 PPWGCN在低训练集比例时的效果评估

### 4.3.1 动机与实验步骤

研究者们发现图卷积神经网络在很小的训练集比例的情况下也能有不错的表现。因为相对于人工智能领域软件工程社区是缺乏高质量标注数据的，所以这个特性对于软件工程社区很有帮助。对于优先级预测，相当多的缺陷报告是没有相应的标签的。所以图卷积神经网络的这个特性可以帮助到优先级自动生成任务。因此作者设计出这个实验以验证PPWGCN是否具有这个特性。

为了验证PPWGCN是否能在很小的训练集比例的情况下也能有不错的表现。作者设置九个不同的训练集比例（0.9、0.8、0.7、0.6、0.5、0.4、0.3、0.2和0.1）在四个项目中部署PPWGCN。

### 4.3.2 结果与分析

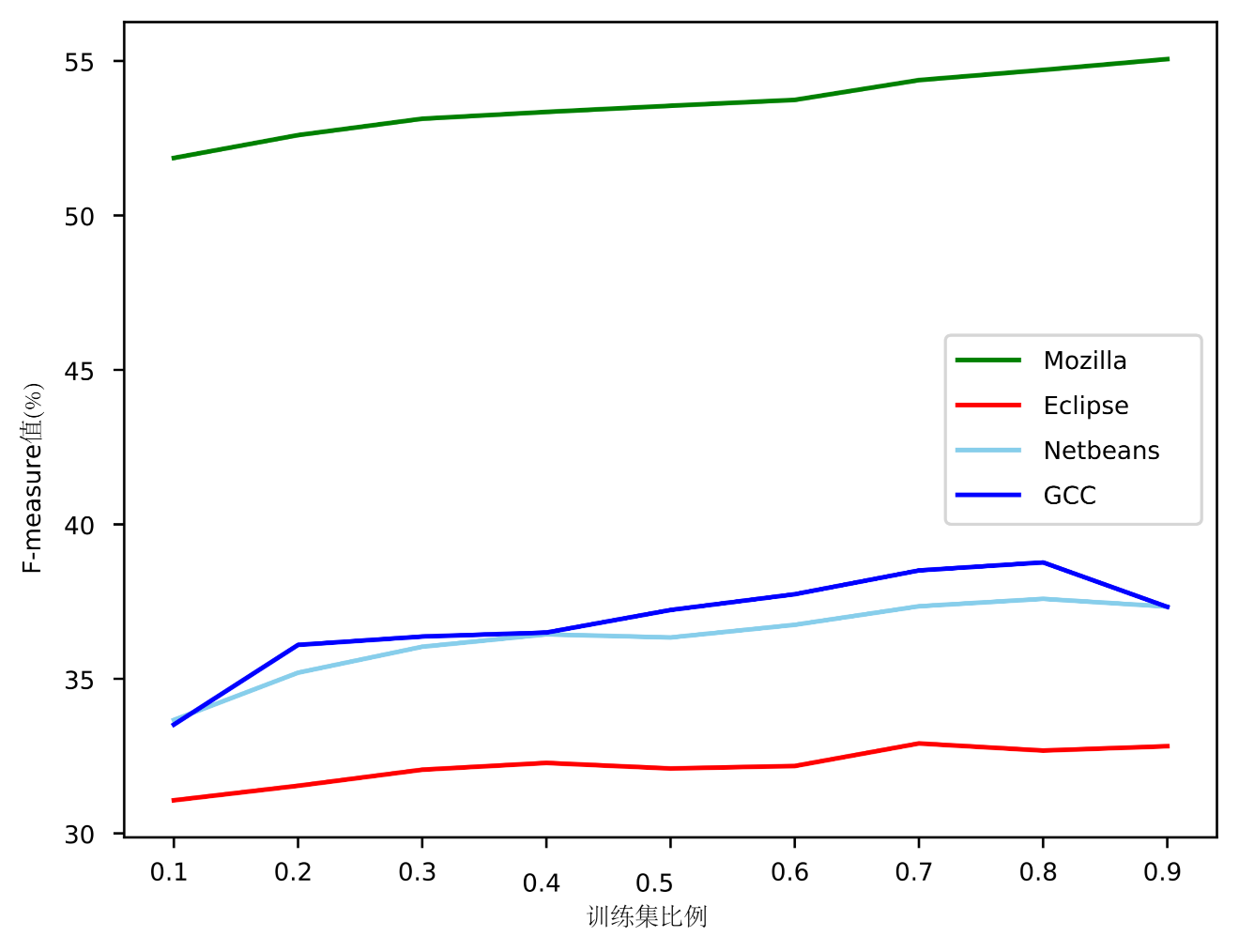
表4.6、表4.7、表4.8、表4.9、表4.10和图4.1展示了不同训练集比例下的结果，

图4.1 不同训练集比例下的结果

表4.6 0.9~0.7训练集比例下F-measure值的结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 项目名称 | 优先级标签 | 0.9(%) | 0.8(%) | 0.7(%) |
| Mozilla | P1 | 65.49 | 67.54 | 66.86 |
| P2 | 36.49 | 35.67 | 35.68 |
| P3 | 61.62 | 61.79 | 61.55 |
| P4 | 46.22 | 45.54 | 43.86 |
| P5 | 65.49 | 63.00 | 63.93 |
| 平均值 | | 55.06 | 54.71 | 54.38 |
| Eclipse | P1 | 24.48 | 23.86 | 24.50 |
| P2 | 25.47 | 25.36 | 25.31 |
| P3 | 76.68 | 76.03 | 75.76 |
| P4 | 16.42 | 15.92 | 16.18 |
| P5 | 21.06 | 22.22 | 22.78 |
| 平均值 | | 32.82 | 32.68 | 32.91 |
| Netbeans | P1 | 51.52 | 50.10 | 50.17 |
| P2 | 49.04 | 50.32 | 50.54 |
| P3 | 60.86 | 61.52 | 58.68 |
| P4 | 25.27 | 26.03 | 27.36 |
| P5 | 0 | 0 | 0 |
| 平均值 | | 37.34 | 37.59 | 37.35 |
| GCC | P1 | 37.45 | 37.11 | 36.47 |
| P2 | 61.30 | 62.08 | 62.36 |
| P3 | 66.16 | 66.81 | 65.52 |
| P4 | 12.33 | 14.71 | 17.54 |
| P5 | 9.41 | 13.17 | 10.68 |
| 平均值 | | 37.33 | 38.77 | 38.51 |

初步可以看出实验结果令人满意。详细分析如下：

（1）对于Mozilla项目，算法在不同训练集比例0.9、0.8、0.7、0.6、0.5、0.4、

0.3、0.2和0.1下的F-measure值分别为55.06%、54.71%、54.38%、53.74%、53.55%、53.35%、53.13%、52.60%和51.86%。

作者发现当训练集比例下降的时候平均F-measure值也随之缓慢下降。作为对比，两个基线算法的平均F-measure值在训练集比例为0.5情况下为24.95%和13.14%。算法在0.1训练集比例情况下可以分别超过两个基线算法26.91%和38.72%。训练集比例为0.1时的平均F-measure只比训练集比例为0.9时的低3.20%，这充分说明了PPWGCN

表4.7 0.6~0.4训练集比例下F-measure值的结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 项目名称 | 优先级标签 | 0.6(%) | 0.5(%) | 0.4(%) |
| Mozilla | P1 | 65.81 | 66.59 | 66.12 |
| P2 | 34.38 | 33.84 | 32.78 |
| P3 | 60.83 | 60.17 | 59.89 |
| P4 | 43.36 | 43.62 | 43.28 |
| P5 | 64.30 | 63.52 | 64.66 |
| 平均值 | | 53.74 | 53.55 | 53.35 |
| Eclipse | P1 | 23.50 | 22.49 | 23.25 |
| P2 | 24.89 | 24.08 | 23.81 |
| P3 | 75.03 | 76.58 | 76.24 |
| P4 | 16.11 | 15.87 | 16.88 |
| P5 | 21.36 | 21.47 | 21.20 |
| 平均值 | | 32.18 | 32.10 | 32.28 |
| Netbeans | P1 | 50.13 | 49.02 | 49.43 |
| P2 | 48.34 | 48.73 | 47.44 |
| P3 | 59.08 | 58.40 | 59.47 |
| P4 | 26.22 | 25.55 | 25.84 |
| P5 | 0 | 0 | 0 |
| 平均值 | | 36.75 | 36.34 | 36.44 |
| GCC | P1 | 37.19 | 36.41 | 37.67 |
| P2 | 59.61 | 59.73 | 58.73 |
| P3 | 66.17 | 64.87 | 62.16 |
| P4 | 16.42 | 15.51 | 15.82 |
| P5 | 9.34 | 9.64 | 8.14 |
| 平均值 | | 37.74 | 37.23 | 36.50 |

可以充分地提取出数据中的有效信息。

（2）对于Eclipse项目，算法在不同训练集比例0.9、0.8、0.7、0.6、0.5、0.4、0.3、0.2和0.1下的F-measure值分别为32.82%、32.68%、32.91%、32.18%、32.10%、 32.28%、32.06%、31.54%和31.07%。

作者发现当训练集比例下降的时候平均F-measure值也随之缓慢下降，虽然在其中有两处增长但是整体还是呈下降趋势的。作为对比，两个基线算法的平均F-measure值在训练集比例为0.5情况下为12.01%和17.92%。算法在0.1训练集比例情况下可以分别超过两个基线算法19.06%和13.15%。训练集比例为0.1时的平均F-measure只比训练

表4.8 0.3~0.1训练集比例下F-measure值的结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 项目名称 | 优先级标签 | 0.3(%) | 0.2(%) | 0.1(%) |
| Mozilla | P1 | 66.14 | 65.30 | 66.64 |
| P2 | 32.66 | 32.48 | 29.28 |
| P3 | 59.51 | 59.35 | 58.37 |
| P4 | 43.37 | 42.95 | 41.63 |
| P5 | 63.96 | 62.94 | 63.38 |
| 平均值 | | 53.13 | 52.60 | 51.86 |
| Eclipse | P1 | 21.66 | 21.25 | 19.83 |
| P2 | 23.78 | 20.99 | 21.54 |
| P3 | 78.33 | 80.33 | 80.45 |
| P4 | 15.18 | 14.67 | 14.41 |
| P5 | 21.36 | 20.46 | 19.13 |
| 平均值 | | 32.06 | 31.54 | 31.07 |
| Netbeans | P1 | 47.99 | 47.16 | 45.70 |
| P2 | 47.85 | 48.00 | 45.85 |
| P3 | 59.89 | 57.76 | 55.23 |
| P4 | 24.48 | 23.07 | 21.59 |
| P5 | 0 | 0 | 0 |
| 平均值 | | 36.04 | 35.20 | 33.67 |
| GCC | P1 | 36.28 | 35.20 | 33.67 |
| P2 | 57.80 | 56.72 | 54.37 |
| P3 | 64.23 | 64.24 | 62.42 |
| P4 | 16.20 | 15.50 | 11.81 |
| P5 | 7.34 | 8.86 | 5.32 |
| 平均值 | | 36.37 | 36.10 | 33.52 |

集比例为0.9时的低1.85%，这充分说明了PPWGCN可以充分地提取出数据中的有效信息。

（3）对于Netbeans项目，算法在不同训练集比例0.9、0.8、0.7、0.6、0.5、0.4、0.3、0.2和0.1下的F-measure值分别为37.34%、37.59%、37.35%、36.75%、36.34%、36.44%、36.04%、35.20%和33.67%。

作者发现当训练集比例下降的时候平均F-measure值也随之缓慢下降，虽然在其中有两处增长但是整体还是呈下降趋势的。作为对比，两个基线算法的平均F-measure值在训练集比例为0.5情况下为27.07%和21.24%。算法在0.1训练集比例情况下可以分

表4.9 0.9~0.5训练集比例下的精确率

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 项目名称 | 优先级标签 | 0.9(%) | 0.8(%) | 0.7(%) | 0.6(%) | 0.5(%) |
| Mozilla | P1 | 77.89 | 76.39 | 76.53 | 76.93 | 76.22 |
| P2 | 31.17 | 31.95 | 31.22 | 29.98 | 30.82 |
| P3 | 57.78 | 57.53 | 58.39 | 57.63 | 56.19 |
| P4 | 39.43 | 41.24 | 37.90 | 35.92 | 37.00 |
| P5 | 62.71 | 59.64 | 62.55 | 63.17 | 61.27 |
| 平均值 | | 53.80 | 53.35 | 53.32 | 52.73 | 52.30 |
| Eclipse | P1 | 17.26 | 17.63 | 18.79 | 18.05 | 16.70 |
| P2 | 22.31 | 19.61 | 19.92 | 18.52 | 19.34 |
| P3 | 88.93 | 89.29 | 89.28 | 89.30 | 88.89 |
| P4 | 11.15 | 11.41 | 10.34 | 10.77 | 10.85 |
| P5 | 14.07 | 15.02 | 16.07 | 15.13 | 15.10 |
| 平均值 | | 30.74 | 30.59 | 30.88 | 30.35 | 30.18 |
| Netbeans | P1 | 51.16 | 49.27 | 49.96 | 47.91 | 47.98 |
| P2 | 53.32 | 53.13 | 51.77 | 52.07 | 51.12 |
| P3 | 62.20 | 62.21 | 63.74 | 61.62 | 62.52 |
| P4 | 18.41 | 20.86 | 19.86 | 19.71 | 18.79 |
| P5 | 0 | 0 | 0 | 0 | 0 |
| 平均值 | | 37.02 | 37.09 | 37.07 | 36.26 | 36.08 |
| GCC | P1 | 30.67 | 31.41 | 33.03 | 32.46 | 31.31 |
| P2 | 63.95 | 63.82 | 61.85 | 64.13 | 61.88 |
| P3 | 71.53 | 71.54 | 70.68 | 68.31 | 68.99 |
| P4 | 8.41 | 10.26 | 12.09 | 10.99 | 10.65 |
| P5 | 6.67 | 9.32 | 7.21 | 6.13 | 6.72 |
| 平均值 | | 36.25 | 37.27 | 36.97 | 36.40 | 35.91 |

别超过两个基线算法6.60%和12.43%。训练集比例为0.1时的平均F-measure只比训练集比例为0.9时的低3.67%，这充分说明了PPWGCN可以充分地提取出数据中的有效信息。

（4）对于GCC项目，算法在不同训练集比例0.9、0.8、0.7、0.6、0.5、0.4、0.3、0.2和0.1下的F-measure值分别为37.33%、38.77%、38.51%、37.74%、37.23%、36.50%、36.37%、36.10%和33.52%。

作者发现当训练集比例下降的时候平均F-measure值也随之缓慢下降，虽然在其中有一处增长但是整体还是呈下降趋势的。作为对比，两个基线算法的平均F-measure值

表4.10 0.4~0.1训练集比例下的精确率

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 项目名称 | 优先级标签 | 0.4(%) | 0.3(%) | 0.2(%) | 0.1(%) |
| Mozilla | P1 | 76.19 | 75.37 | 75.05 | 72.07 |
| P2 | 30.76 | 30.13 | 29.27 | 29.13 |
| P3 | 54.42 | 54.61 | 53.94 | 52.57 |
| P4 | 35.72 | 37.32 | 38.24 | 39.02 |
| P5 | 63.81 | 61.53 | 61.09 | 61.78 |
| 平均值 | | 52.18 | 51.79 | 51.52 | 50.91 |
| Eclipse | P1 | 17.63 | 17.20 | 16.25 | 16.39 |
| P2 | 19.77 | 18.92 | 18.95 | 18.46 |
| P3 | 88.97 | 88.18 | 87.14 | 86.53 |
| P4 | 11.11 | 10.86 | 11.12 | 10.86 |
| P5 | 14.04 | 15.45 | 15.03 | 14.99 |
| 平均值 | | 30.31 | 30.12 | 29.70 | 29.44 |
| Netbeans | P1 | 47.24 | 47.49 | 46.03 | 45.43 |
| P2 | 51.86 | 51.19 | 49.23 | 47.67 |
| P3 | 61.56 | 60.74 | 61.35 | 60.70 |
| P4 | 19.22 | 18.69 | 17.86 | 15.12 |
| P5 | 0 | 0 | 0 | 0 |
| 平均值 | | 35.98 | 35.62 | 34.89 | 33.78 |
| GCC | P1 | 31.10 | 31.43 | 30.93 | 30.62 |
| P2 | 60.67 | 60.70 | 61.04 | 56.68 |
| P3 | 69.09 | 67.16 | 65.87 | 63.97 |
| P4 | 10.05 | 10.81 | 10.31 | 8.12 |
| P5 | 5.26 | 5.32 | 6.11 | 4.03 |
| 平均值 | | 35.23 | 35.08 | 34.85 | 32.68 |

在训练集比例为0.5情况下为34.02%和13.12%。算法在0.2训练集比例情况下可以分别超过两个基线算法2.08%和22.98%。训练集比例为0.1时的平均F-measure只比训练集比例为0.9时的低3.81%，这充分说明了PPWGCN可以充分地提取出数据中的有效信息。

尽管算法只构建出一张包含所有缺陷报告的图，但是它可以通过图卷积神经网络的两次传播充分提取出全局语义信息。所以实验表明PPWGCN可以在很低训练集比例的情况下获得很好的效果。

## 4.4 加权损失函数解决数据不平衡问题的验证

### 4.4.1 动机与实验步骤

同现存研究一样，作者的数据非常不平衡。不平衡的数据会给实验的结果造成很大的影响。具体来说，机器学习分类模型倾向于朝着数量多的标签的方向学习，所以数量多的标签的准确率接近百分之百。同时，那些数量少的标签的准确率接近为0。为了解决这个问题，作者在损失函数中引入了标签罚项。为了验证方法是否有效，作者设计了这个实验。

为了验证引入标签罚项是否可以解决数据不平衡问题，作者对使用普通损失函数的PPWGCN进行实验，方法被称为PPGL（the Priority Prediction Algorithm with General Loss function）。

### 4.4.2 结果与分析

图4.2、图4.3、图4.4、图4.5和表4.11展示了PPWGCN和PPGL的实验结果。详细分析如下：

（1）对于Mozilla项目，PPGL对五个类别标签P1、P2、P3、P4和P5在F-measure值指标下的预测结果为73.45%、24.97%、59.54%、46.14%和68.16%。其中本文提出的算法对于P1标签的预测效果最好，P2的效果最差。PPWGCN的平均F-measure值为54.45%。

作者使用方差指标判断加权损失函数是否能够解决数据不平衡问题。PPWGCN对于 P1、P2、P3、P4和P5标签的预测F-measure值分别为66.59%、33.84%、60.17%、43.62%和63.52%。PPWGCN和PPGL方差结果分别为0.0200和0.0378，所以PPWGCN的结果更加平衡。

（2）对于Eclipse项目，PPGL对五个类别标签P1、P2、P3、P4和P5在F-measure值指标下的预测结果为17.05%、17.94%、87.51%、11.76%和19.50%。其中本文提出的算法对于P1标签的预测效果最好，P2的效果最差。PPWGCN的平均F-measure值为30.75%。

作者使用方差指标判断加权损失函数是否能够解决数据不平衡问题。PPWGCN对于 P1、P2、P3、P4和P5标签的预测F-measure值分别为22.49%、24.08%、76.58%、15.87%和21.47%。PPWGCN和PPGL方差结果分别为0.0628和0.1015，所以PPWGCN的结果更加平衡。

表4.11 PPGL的实验结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 项目名称 | 优先级标签 | 精确率(%) | 召回率(%) | F-measure值(%) |
| Mozilla | P1 | 67.11 | 81.10 | 73.45 |
| P2 | 35.39 | 19.29 | 24.97 |
| P3 | 57.36 | 61.89 | 59.54 |
| P4 | 62.92 | 36.43 | 46.14 |
| P5 | 76.93 | 61.19 | 68.16 |
| 平均值 | | 59.94 | 51.98 | 54.45 |
| Eclipse | P1 | 24.22 | 13.15 | 17.05 |
| P2 | 22.37 | 14.97 | 17.94 |
| P3 | 83.84 | 91.51 | 87.51 |
| P4 | 18.46 | 8.63 | 11.76 |
| P5 | 25.37 | 15.84 | 19.50 |
| 平均值 | | 34.85 | 28.82 | 30.75 |
| Netbeans | P1 | 45.85 | 45.27 | 45.56 |
| P2 | 49.24 | 48.06 | 48.64 |
| P3 | 56.45 | 62.46 | 59.30 |
| P4 | 26.32 | 13.92 | 18.21 |
| P5 | 0 | 0 | 0 |
| 平均值 | | 35.57 | 33.94 | 34.34 |
| GCC | P1 | 31.11 | 27.04 | 28.93 |
| P2 | 58.39 | 62.51 | 60.38 |
| P3 | 66.12 | 65.73 | 65.92 |
| P4 | 18.18 | 11.46 | 14.06 |
| P5 | 6.90 | 3.31 | 4.47 |
| 平均值 | | 36.14 | 34.01 | 34.75 |

（3）对于Netbeans项目，PPGL对五个类别标签P1、P2、P3、P4和P5在F-measure值指标下的预测结果为45.56%、48.64%、59.30%、18.21%和0%。其中本文提出的算法对P1标签的预测效果最好，P2效果最差。PPWGCN的平均F-measure值为34.34%。

作者使用方差指标判断加权损失函数是否能够解决数据不平衡问题。PPWGCN对于 P1、P2、P3、P4和P5标签的预测F-measure值分别为49.02%、48.73%、58.40%、25.55%和0%。PPWGCN和PPGL方差结果分别为0.0559和0.0598，所以PPWGCN的结果更加平衡。

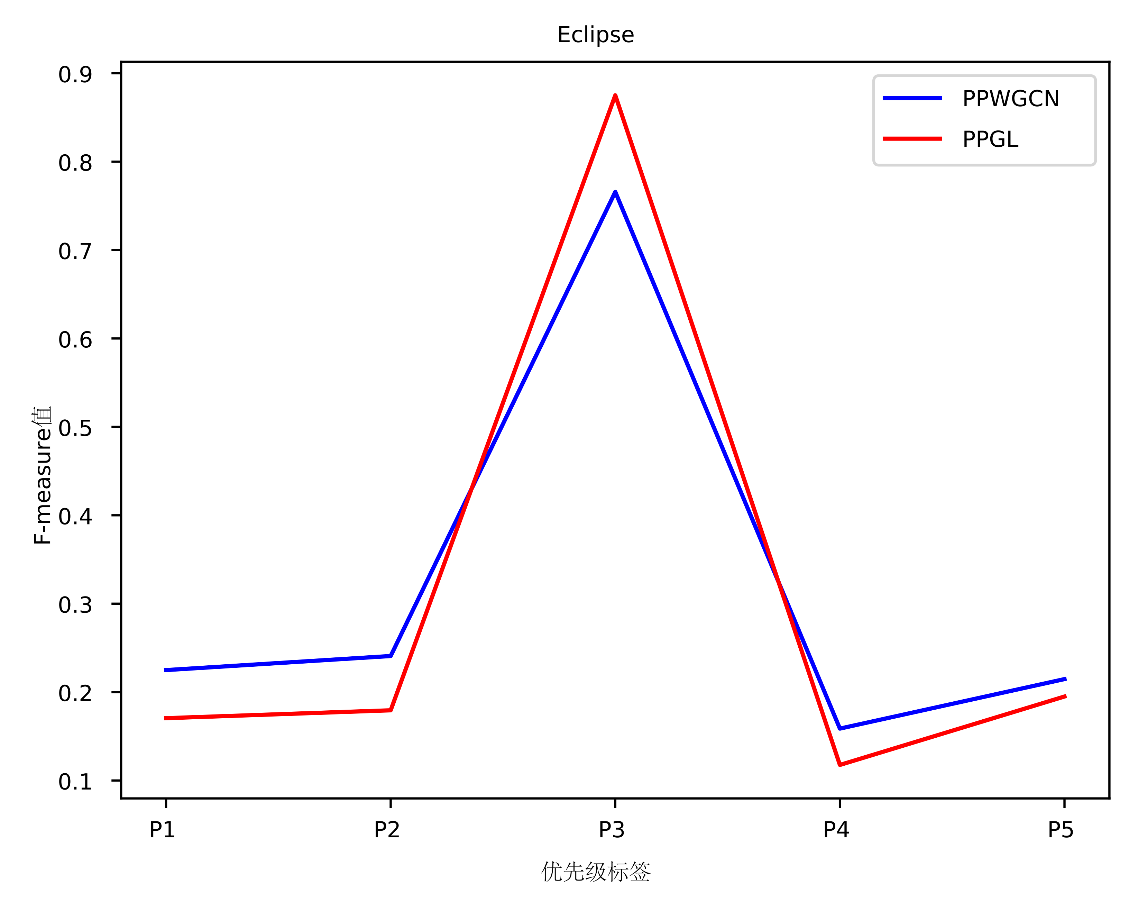
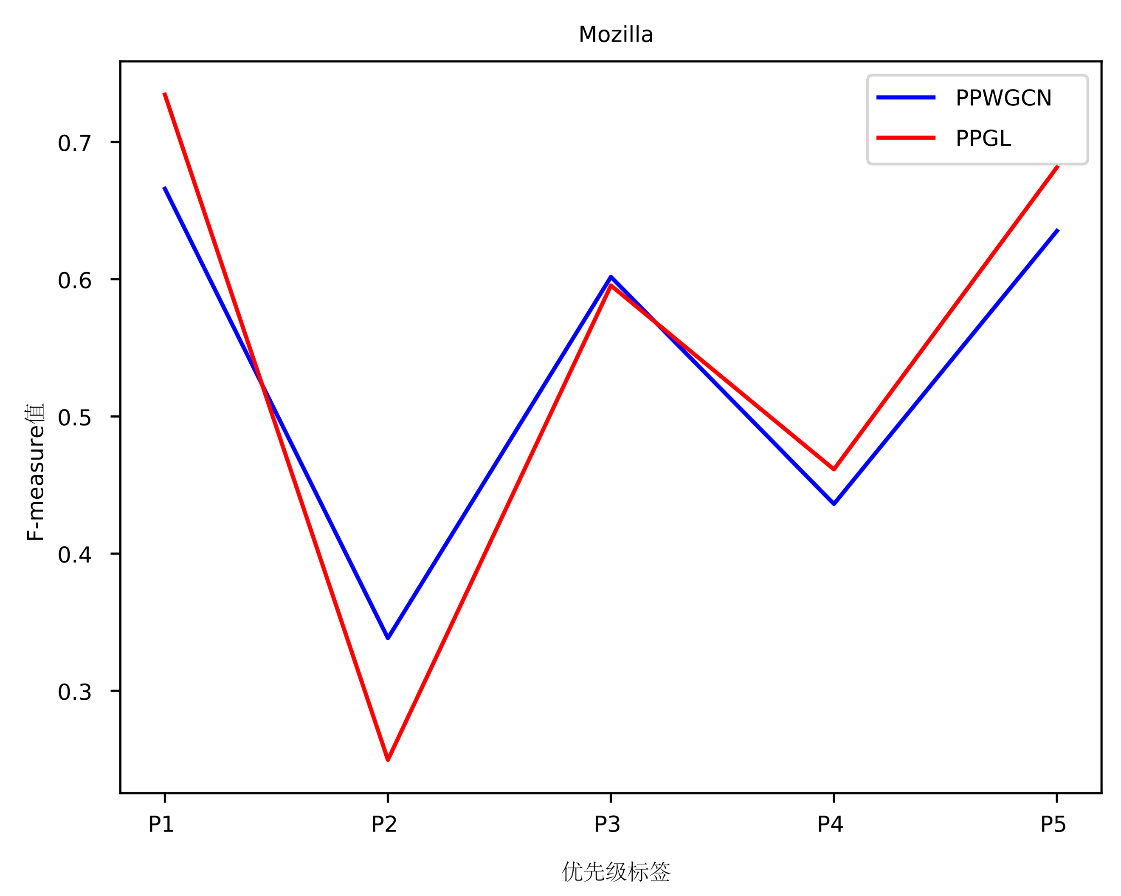
图4.2 Mozilla项目中两种算法的对比

图4.3 Eclipse项目中两种算法的对比

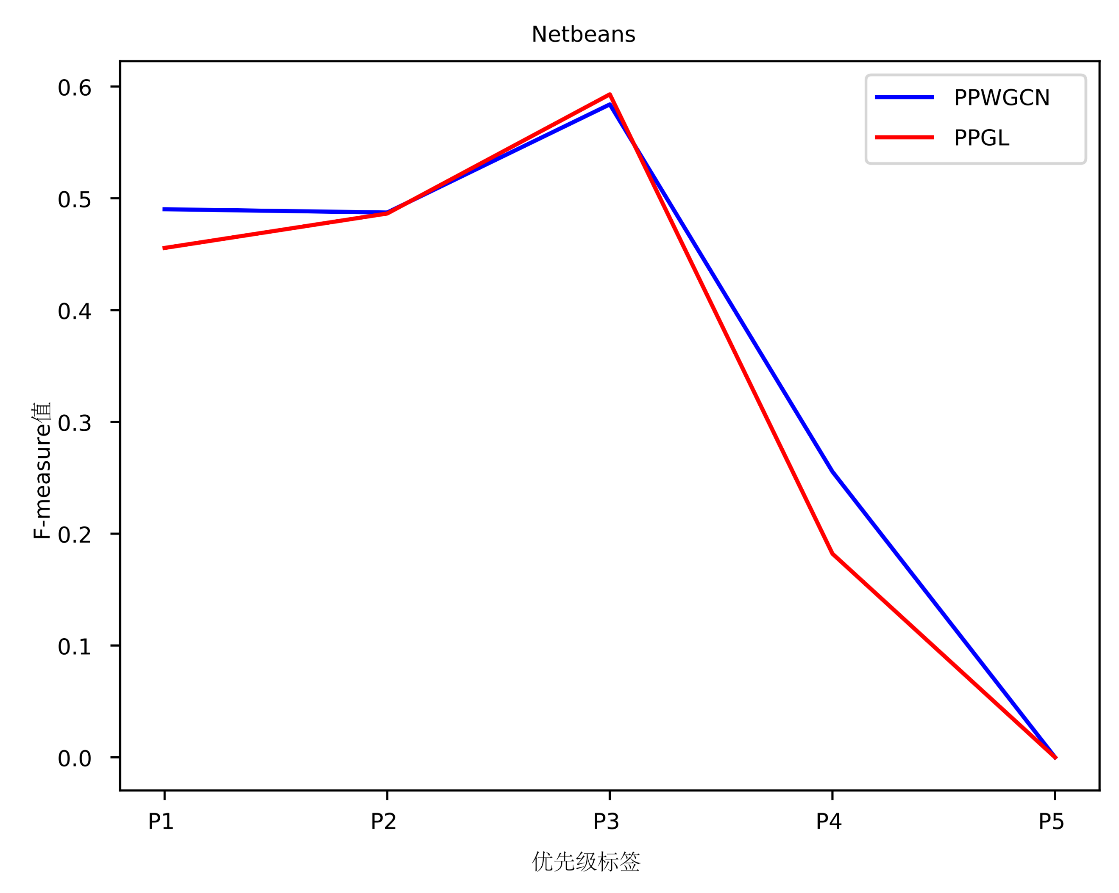
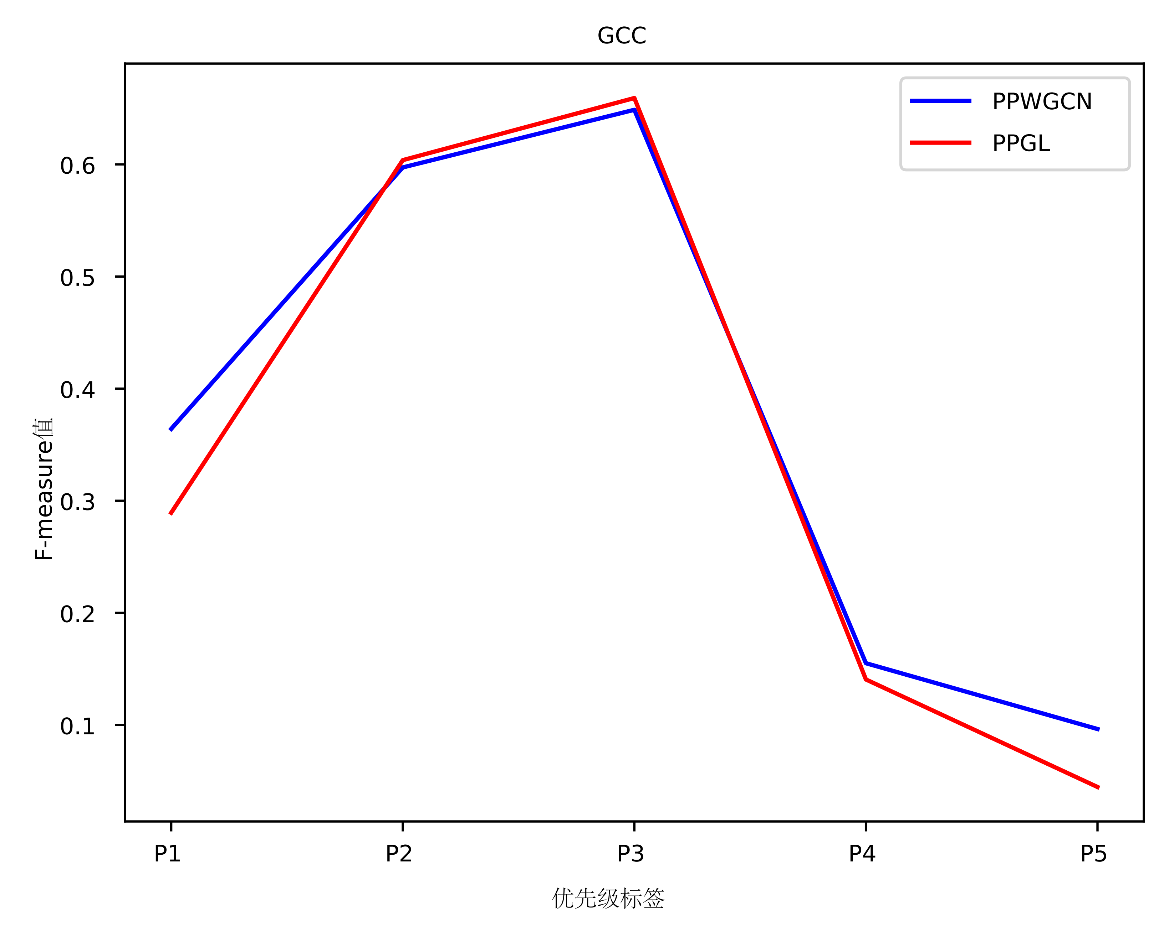
图4.4 Netbeans项目中两种算法的对比

图4.5 GCC项目中两种算法的对比

（4）对于GCC项目，PPGL对五个类别标签P1、P2、P3、P4和P5在F-measure值指标下的预测结果为28.93%、60.38%、65.92%、14.06%和4.47%。其中本文提出的算法对P1标签的预测效果最好，P2的效果最差。PPWGCN的平均F-measure值为34.75%。

作者使用方差指标判断加权损失函数是否能够解决数据不平衡问题。PPWGCN对于P1、P2、P3、P4和P5标签的预测F-measure值分别为36.41%、59.73%、64.87%、15.51%和9.64%。PPWGCN和PPGL方差结果分别为0.0626和0.0752，所以PPWGCN的结果更加平衡。

综上所述引入标签罚项是可以应对数据不平衡问题的。

## 4.5 本章小结

本章首先介绍了实验中用到的数据、评价指标和参数设置。然后从动机到实验步骤，再到结果与分析详细描述精心设计的三个实验。第一个实验是为了对比PPWGCN与两个基线算法以验证其效果，结果表明其显著地超过两个基线算法。第二个实验是为了评估PPWGCN在低训练集比例下的效果，结果表明其在低训练集比例下依然能够表现良好。第三个实验是为了验证PPWGCN中的加权损失函数是否可以解决数据不平衡问题，实验表明这样的设置可以应对数据不平衡问题。

# 结 论

随着越来越多的软件开发团队开始使用缺陷追踪系统，科研人员希望可以使得缺陷报告生命周期中某些环节实现自动化，其中就包括对缺陷报告优先级标注的自动化。但现存算法存在着不足，所以本文提出一个基于图卷积神经网络的缺陷报告优先级自动生成算法PPWGCN。

本文的贡献如下：

（1）设计出一个基于图卷积神经网络并设置加权损失函数的缺陷报告优先级自动生成算法PPWGCN。为了验证算法的效果，选取两个优秀的现存方法作为基线算法，DRONE和cPur。首先复现两个基线算法，DRONE利用线性回归并训练出四个阈值以实现缺陷报告优先级的自动生成；cPur结合情感分析使用卷积神经网络实现缺陷报告优先级的自动生成。然后在爬取的四个开源项目中的258012份缺陷报告上运行这三个算法。最后实验表明：PPWGCN在平均F-measure指标上超过两个基线算法120.89%，且结果相对平衡。

（2）由于软件工程领域缺乏优质的数据，所以如果某个算法在低训练集比例的情况下也能表现良好，那么它将会非常适用于软件工程领域的任务。为了验证PPWGCN是否具有这个特性，作者在数据集中使用不同训练集比例（0.9、0.8、0.7、0.6、0.5、0.4、0.3、0.2和0.1）运行PPWGCN，结果表明PPWGCN在训练集比例为0.1时仍有不错的效果。

（3）为了应对数据不平衡问题，作者设计了一个加权损失函数。为了验证这个设计能否解决问题，作者使用普通损失函数在数据集上进行实验。实验表明PPWGCN结果的方差大于使用普通损失函数时结果的方差。所以PPWGCN中的加权损失函数可以应对数据不平衡问题。

作者只选取了Bugzilla系统中的开源项目的缺陷报告作为实验数据，而不确定PPWGCN在商业项目或者其他缺陷追踪系统中能否依然表现良好。所以未来将尝试在其他项目中进行实验以验证PPWGCN。鉴于图卷积神经网络在低训练集比例时的表现依然令人满意且训练速度快，作者将尝试使用图卷积神经网络完成软件工程领域其他的任务：缺陷摘要、代码搜索和副本探测等等。

# 参考文献

[1] Tian Y, Lo D, Xia X, et al. Automated prediction of bug report priority using multi-factor analysis[J]. Empirical Software Engineering, 2015, 20(5): 1354-1383.

[2] Alenezi M, Banitaan S. Bug reports prioritization: Which features and classifier to use?[A]. International Conference on Machine Learning and Applications[C]. Miami: IEEE, 2013: 112-116.

[3] Umer Q, Liu H, Illahi I. CNN-based automatic prioritization of bug reports[J]. IEEE Transactions on Reliability, 2019, 69(4): 1341-1354.

[4] Abdelmoez W, Kholief M, Elsalmy F M. Bug fix-time prediction model using naïve bayes classifier[A]. International Conference on Computer Theory and Applications[C]. Alexandria: IEEE, 2012: 167-172.

[5] Choudhary P A, Singh S. Neural Network Based Bug Priority Prediction Model using Text Classification Techniques[J]. International Journal of Advanced Research in Computer Science, 2017, 8(5): 1-13.

[6] Menzies T, Marcus A. Automated severity assessment of software defect reports[A]. International Conference on Software Maintenance[C]. Beijing: IEEE, 2008: 346-355.

[7] Lamkanfi A, Demeyer S, Giger E, et al. Predicting the severity of a reported bug[A]. Working Conference on Mining Software Repositories[C]. Cape Town: IEEE, 2010: 1-10.

[8] Tan Y, Xu S, Wang Z, et al. Bug severity prediction using question-and-answer pairs from stack overflow[J]. Journal of Systems and Software, 2020, 165(1): 110-145.

[9] 刘文杰. 软件缺陷报告严重性预测研究[D]. 大连: 大连理工大学, 2020.

[10] Sabor K K, Hamdaqa M, Hamou-Lhadj A. Automatic prediction of the severity of bugs using stack traces and categorical features[J]. Information and Software Technology, 2020, 123(1): 106-157.

[11] Yao L, Mao C, Luo Y. Graph convolutional networks for text classification[A]. AAAI Conference on Artificial Intelligence[C]. Honolulu: AAAI, 2019: 7370-7377.

[12] 夏鑫, 王新宇, 杨小虎, 罗大卫. 开源软件系统缺陷报告管理与分析[J]. 中国计算机学会通讯, 2016, 12(2): 29-34.

[13] 喻维. 软件缺陷报告管理关键技术研究[D]. 广州: 华南理工大学, 2018.

[14] Zhang T, Chen J, Yang G, et al. Towards more accurate severity prediction and fixer recommendation of software bugs[J]. Journal of Systems and Software, 2016, 117(1): 166-184.

[15] Minaee S, Kalchbrenner N, Cambria E, et al. Deep Learning--based Text Classification: A Comprehensive Review[J]. ACM Computing Surveys, 2021, 54(3): 1-40.

[16] Kalchbrenner N, Grefenstette E, Blunsom P, et al. A Convolutional Neural Network for Modelling Sentences[A]. Annual Meeting of the Association for Computational Linguistics[C]. Baltimore: The Association for Computer Linguistics, 2014: 212-217.

[17] Zhao W, Ye J, Yang M, et al. Investigating capsule networks with dynamic routing for text classification[A]. Conference on Empirical Methods in Natural Language Processing[C]. Brussels: Association for Computational Linguistics, 2018: 3110-3119.

[18] Yang M, Zhao W, Chen L, et al. Investigating the transferring capability of capsule networks for text classification[J]. Neural Networks, 2019, 118(3): 247-261.

[19] Zhao W, Peng H, Eger S, et al. Towards Scalable and Reliable Capsule Networks for Challenging NLP Applications[A]. Annual Meeting of the Association for Computational Linguistics[C]. Florence: Association for Computational Linguistics, 2019: 1549-1559.

[20] Shen T, Zhou T, Long G, et al. Disan: Directional self-attention network for rnn/cnn-free language understanding[A]. AAAI Conference on Artificial Intelligence[C]. New Orleans: AAAI, 2018: 5446-5455.

[21] Munkhdalai T, Yu H. Neural semantic encoders[A]. Conference of the European Chapter of the Association for Computational Linguistics[C].Valencia: Association for Computational Linguistics, 2017: 397-407.

[22] Devlin J, Chang M W, Lee K, et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding[A]. Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies[C]. Minneapolis: Association for Computational Linguistics, 2019: 4171-4186.

[23] Wu Z, Pan S, Chen F, et al. A comprehensive survey on graph neural networks[J]. IEEE transactions on neural networks and learning systems, 2020, 32(1): 4-24.

[24] Li Y, Yu R, Shahabi C, et al. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting[A]. International Conference on Learning Representations[C]. Vancouver: OpenReview.net, 2018: 1-18.

[25] Jain A, Zamir A R, Savarese S, et al. Structural-rnn: Deep learning on spatio-temporal graphs[A]. Conference on Computer Vision and Pattern Recognition[C]. Las Vegas: IEEE Computer Society, 2016: 5308-5317.

[26] Yan S, Xiong Y, Lin D. Spatial temporal graph convolutional networks for skeleton-based action recognition[A]. AAAI conference on artificial intelligence[C]. New Orleans: AAAI Press, 2018: 7444-7452.

[27] Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks[A]. International Conference on Learning Representations[C].Toulon: OpenReview.net, 2017:256-271.

# 攻读学士学位期间发表的论文和取得的科研成果

[1] **Tan Youshuai**, Xu Sijie, Wang Zhaowei, Zhang Tao, Xu Zhou, Luo Xiapu. Bug severity prediction using question-and-answer pairs from Stack Overflow[J]. **Journal of Systems and Software**. 2020, 165: 110567. CCF-B, 中科院二区SCI期刊, SCI收录: 000530186500003, 第一作者.

[2] Fang Sen, **Tan Youshuai**, Zhang Tao, Liu Yepang. Self-Attention Networks for Code Search[J]. **Information and Software Technology**. 2021, 134: 106542. CCF-B, 中科院一区SCI期刊, SCI收录: 000634797600003, 第二作者.

[3] Fang Sen, **Tan Youshuai**, Zhang Tao, Xu Zhou, Liu Hui. Effective Prediction of Bug-Fixing Priority via Weighted Graph Convolutional Networks[J]. **IEEE Transactions on Reliability**. 2021, 70(2): 563-574. 中科院一区SCI期刊, 在线发表, 并列第一作者.

# 致 谢

论文撰写的结束意味着本科的旅途已接近尾声，四年如白驹过隙，有的记忆已经模糊，有的场景却仍历历在目。回首本科生涯，我非常感激每一个曾给过我帮助的人。

我的毕业设计指导老师是刘刚老师，他总能及时解答我的问题，给出专业且明确的建议。他会严格把控自己设置的时间结点以保证我们都能顺利完成毕业设计。考虑到我们是第一次撰写毕业论文，刘老师在写论文前开会详细介绍论文写作的要点和技巧。写作完成后，刘老师也帮每个人详细地修改论文。感谢刘老师对我毕业设计的指导，从他的身上我学到了做科研的严谨。

关于科研，很感激张涛老师在大一下开始带着我做科研，给我一个比较高的起点。同时也很感激我的合作者们，香港理工的罗夏朴老师、康考迪亚大学的尚嵬嶷老师、重庆大学的徐洲老师、澳门科技大学的博士生方森、康考迪亚大学的陈金富博士和哈尔滨工业大学的王昭为同学。他们给了我许多宝贵的建议和指导，希望将来可以一起收获更多。

校园之外，我非常感激我的家人们。他们是我坚实的后盾，使我不必在乎眼前的苟且，全力追逐我的科研梦。

最后感谢母校对我四年的培养，帮助我为梦想安装上翅膀。今后若鹏程万里，定不忘母校之恩。

希望母校将来可以以我为荣。