融合句嵌入模型和代码特征的补丁验证方法

蒋婷婷,姜淑娟,韩 威
(中国矿业大学计算机科学与技术学院,江苏徐州 221116)

摘 要: 补丁验证常用运行测试套件的方法来验证补丁正确性,然而自动修复技术生成的补丁往往数量巨大,而将每个补丁依次通过测试套件则会产生难以承受的开销. 针对该问题,本文提出一个由句嵌入模型 InferSent 和支持向量机分类器组成的静态补丁验证方法. 使用 InferSent 提取代码静态特征并通过支持向量机分类器来预测补丁正确性. 该方法更加关注代码的静态特征信息,通过对特征的提取分析,无需运行测试套件即可有效地预测自动修复工具生成的补丁的正确性. 本文在多个自动修复工具生成的补丁集合上进行了验证. 实验结果表明,在修复工具生成的补丁集合上,本文提出的静态补丁验证方法对补丁预测的 F1 值达到 71.89%,相比其他两种最新静态补丁验证方法分别提高 11.64% 和 6.43%,并在五项评价指标上均优于对比模型. 表明该方法可以在不运行测试套件的情况下正确预测补丁,且具有良好的泛化能力.

关键词: 程序自动修复;补丁验证;代码静态特征;句嵌入技术;支持向量机;代码相似性
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112(2023)12-3450-07
电子学报 URL:http://www.ejournal.org.cn DOI:10.12263/DZXB.20230048

Patch Verification Method Integrating Sentence Embedding Model and Code Features

JIANG Ting-ting, JIANG Shu-juan, HAN Wei
(*School of Computer Science & Technology*, *China University of Mining and Technology*, *Xuzhou*, *Jiangsu* 221116, *China*)

Abstract: Patch verification often runs a test suite to verify the correctness of patches, however, the number of patches generated by automatic repair techniques is often huge, and passing each patch through the test suite in turn incurs an unbearable overhead. To address this problem, this paper proposes a static patch verification method consisting of a sentence embedding model InferSent and an support vector machine (SVM) classifier. InferSent is used to extract static features of the code and the SVM classifier is used to predict the patch correctness. The method focuses more on the static feature information of the code, and by extracting and analyzing the features, it can effectively predict the correctness of the patches generated by automatic repair tools without running a test suite. In this paper, it is validated on a collection of patches generated by several automatic repair tools. The experimental results show that the static patch validation method proposed in this paper achieves an F1 value of 71.89% for patch prediction on the patch sets generated by the repair tool, which is 11.64% and 6.43% higher than the other two state-of-the-art static patch validation methods, respectively, and outperforms the comparison models in terms of all five evaluation metrics. It is shown that the method can correctly predict patches without running the test suite and has good generalization capability.

Key words: program automatic repair; patch verification; code static characteristics; sentence embedding technology; support vector machine; code similarity

1 引言

随着软件工程领域的蓬勃发展,各类软件的使用极大丰富了人类生活. 然而开发人员在开发软件时,常需要花费极大的时间精力去修复程序错误,这对开发和维护软件系统极为不利.

程序自动修复方法自动生成正确的补丁片段来替换错误片段,可以帮助减少开发成本[1]. 补丁验证是自动修复领域的最后一个阶段,即在补丁生成之后确定补丁正确性. 其对最终能否高效、正确地修复错误起到重要作用. 为了在补丁验证阶段减少验证开销,并提高候选补丁的正确率. 许多研究人员进行了补丁正确性

收稿日期:2023-01-13;修回日期:2023-10-07;责任编辑:李勇锋

的研究[2]. 现有的补丁验证技术可以根据是否执行测试用例分为两种类型:静态技术和动态技术. 静态技术通过提取和分析代码特征来预测和过滤补丁. 动态技术则是利用自动化测试生成工具来识别补丁. 研究表明[3]现有的静态代码特征能够有效地区分过拟合补丁和正确的补丁. 分析静态代码特征过滤补丁的验证方法由于不需要运行测试套件,不需要反复编译代码,其开销远低于动态技术.

句嵌入是对文本的每个句子进行编码提取其静态特征的一种方式,在自然语言学习领域广泛应用,在文本分类、情感分析等领域取得了突出的研究成果. 与自然语言一样,代码也是重复的和可预测的[4]. 而在源代码的各种表示中,源代码能够保留原始程序的大部分信息[5],最适合执行确定候选补丁可靠性的任务.

综上本文利用句嵌入模型对错误代码-补丁代码对进行表征学习,将代码片段映射到向量空间,然后计算代码对之间的静态特征,对提取的静态特征进行机器学习训练与分类,通过分类器对候选补丁进行分类,从而验证补丁的正确性,达到在不运行测试套件的情况下提高补丁正确率的目的.

2　背景介绍

2.1　补丁验证

补丁验证技术是自动修复的研究重点,在修复过程中补丁验证阶段消耗时间占总耗时的90%[6]. 因此补丁验证技术的改进对最终能否高效、正确地修复错误起到重要作用. 传统的补丁验证技术每次测试需要重新编译修复程序,以确定是否能够通过所有测试套件,这个过程循环重复直至找到能够通过所有测试套件的补丁或者超时. 然而,自动修复技术整体生成补丁的正确率偏低,传统的补丁验证方法可能消耗大量重编译时间. 因此近些年,随着深度学习技术的进展,一些研究人员考虑利用相关技术在不运行测试套件的情况下解决补丁排序及验证问题[3].

2.2　句嵌入模型

句嵌入技术,即将一个完整的语句映射成一个实数向量的技术. 传统的句嵌入方法常采用无监督学习方法,然而无监督的学习方法在较长语句向量的获得方面表现得不够优异[7]. 因此本文考虑监督学习模型InferSent. 其采用双向长短期记忆网络,并用最大池化操作作为句子的编码器. 该模型使用斯坦福自然语言推断数据集[8]作为训练集,训练集包含57万个人类产生的句子对,每个句子对都被打上标签,代表句子对中两个句子(前提、假设)之间的关系,这与本文区分正确补丁与过拟合补丁的目的相同.

图1左半部分描述了InferSent的预训练框架. 具体

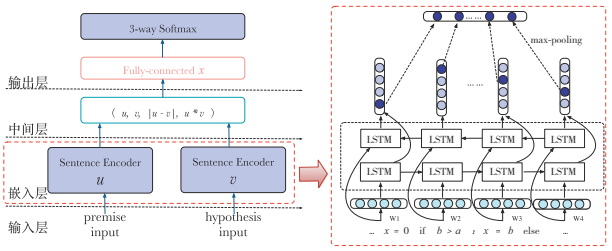


图1　InferSent嵌入模型框架

来说,该框架可分为四层,其中将在中间层提取的特征输入到输出层中,输出层包含全连接层和三分类softmax层,将句子对输出为句子对关系预测值,与数据集已有标签对应,用反向进行数据更新的方式将模型进行优化训练,得到能够较好地提取句子特征的句子编码器. 其预训练结构如图1右半部分所示,包含两个Bi-LSTM和Max-pooling两部分. Bi-LSTM是LSTM的拓展,使用双向的LSTM层,本文中使用它来提取代码的文本特征. 将数据预处理后输入嵌入模型中,对于一组T个单词的句子,双向长短期记忆模型计算一组T向量,它由前向LSTM与后向LSTM串联而成,以两个相反的方向读取句子,然后通过选择隐藏单元的每个维度上的最大值(Max-pooling)组合成固定大小的向量,即为所需的高维代码向量.

3　补丁验证方法

3.1　总体框架

图2展示了本文提出的补丁验证方法框架,分为四个阶段. 第一阶段收集了如表1、表2中所示的用于训练嵌入模型和验证模型效果的补丁集合. 第二阶段,对收集得到的补丁集合进行预处理,将补丁集合处理成错误代码-补丁代码对的单行文本. 对于每个错误代码-补丁代码对,在第三阶段中,使用嵌入模型进行向量映射和特征提取,微调过程如算法1所示. 最后,在第四阶段中用support vector machine (SVM)分类器训练代码对的静态特征,并对补丁正确性进行预测.

3.2　代码预处理方法

本文使用的补丁代码源文件示例如图3所示,本文对补丁出现的代码片段及其上下文进行预处理. 图3包含需要删除的代码buggy和需要添加的代码patched的原始补丁文件. 其中,需要被删除的行以‘—’号开头,需要被添加的行以‘+’号开头,其余行为补丁上下文行. 将补丁数据集中全部补丁p中每个原始补丁文件生成为单行三元组〈label,buggy,patched〉,按行依次写入列表d,最终将包含全部补丁处理完的三元组列表d写入以“.txt”为后缀名的文件C,以方便后续进行补丁代码嵌入与训练验证.

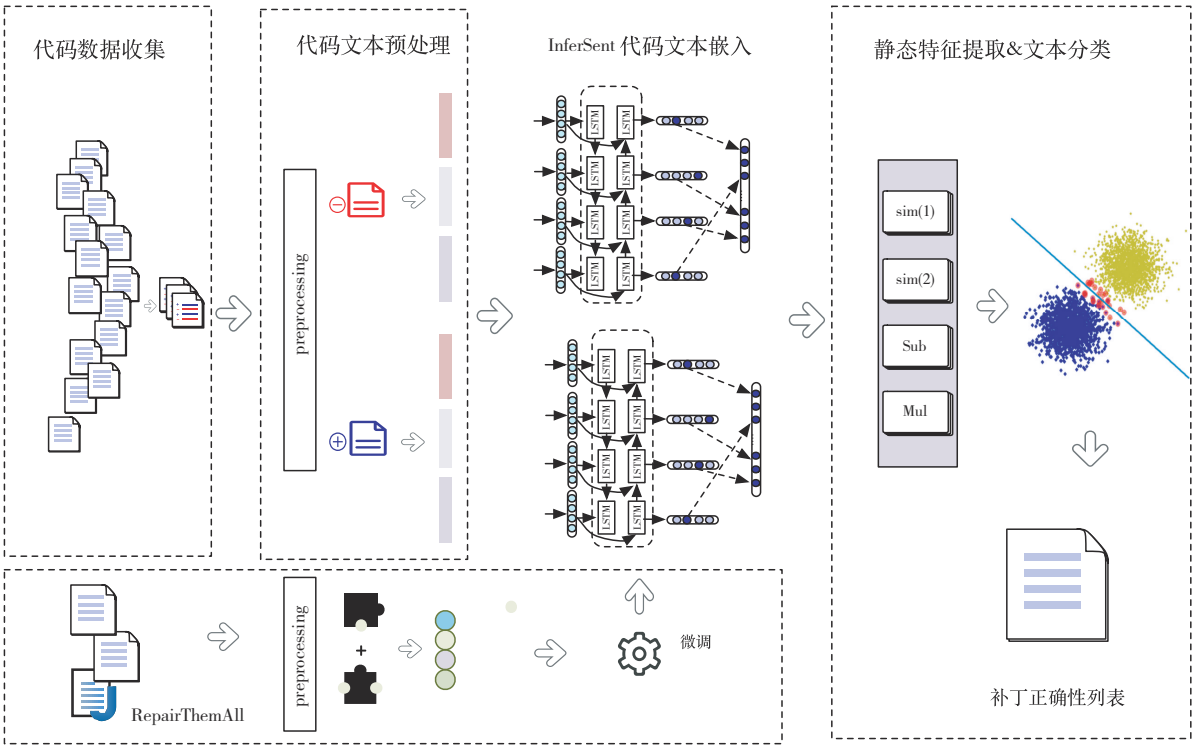


图2 本文方法的整体框架

算法1 Infersent模型的微调算法

输入:基准数据集合 C

输出:预训练后的Bi-LSTM编码器

初始化batch_size,迭代次数Epoch,嵌入维度 D;

FOR each batch in C DO

初始化空向量 e_b 和 e_p;

FOR each buggy and patch in batch DO

前向 LSTM_L 计算buggy和patch的前向编码 b_L 和 p_L;

后向 LSTM_R 计算buggy和patch的后向编码 b_R 和 p_R;

拼接buggy和patch的前向和后向编码得到 b 和 p;

对 b 和 p 使用Max-pooling得到 b_m 和 p_m;

将编码向量 b 和 p 分别写入 e_b 和 e_p;

END FOR

使用 e_b 和 e_p 计算相减特征 E_s、相乘特征 E_m、余弦相似度 E_c 以及欧几里得相似度 E_e;

拼接特征 $[E_s;E_m;E_c;E_e]$;

全连接层特征映射;

交叉熵损失函数计算损失值;

反向传播更新Bi-LSTM参数;

END FOR

3.3 模型训练

句嵌入是句子的高维表示,为了能够更好地提取代码文本特征,本文采用监督学习预训练的句子编码器InferSent进行文本嵌入. 本文首先使用文献[7]中根据SNLI文本数据集[8]预训练好的Bi-LSTM编码器参数. 针对该编码器的参数使用代码领域的知识(RepairThemAll数据集[9])对其进行微调,其由11个自动修复工具生成的64 293个补丁组成. 由于微调需要有标签数据,本文采用Tian等人[10]按照Liu等人[11]列举的严格标准标记的RepairThemAll数据集. 原本数据集中的句子对被替换做patched和buggy代码对作为数据输入. 由于是微调训练,本文将学习率设置为一个很小的数值(1×10^{-5}). 最后将微调得到的Bi-LSTM编码器作为代码嵌入工具. 微调过程见算法1. 通过不断完成前向计算和反向传播,InferSent模型的编码器(Bi-LSTM)能够学习到合适的权重参数. 本文保存得到的编码器参数,并将其迁移到补丁分类任务中,从而进行代码嵌入,最后计算补丁的正确性.

```
diff --git a/src/com/google/javascript/jscomp/Compiler.java b/src/com/google/javascript/jscomp/Compiler.java
index 3756b99..8898f77 100644
--- a/src/com/google/javascript/jscomp/Compiler.java
+++ b/src/com/google/javascript/jscomp/Compiler.java
@@ -1285,7 +1285,7 @@ public class Compiler extends AbstractCompiler {

     // Check if the sources need to be re-ordered.
     boolean staleInputs = false;
-    if (options.dependencyOptions.needsManagement()) {
+    if (options.dependencyOptions.needsManagement() && options.closurePass) {
       for (CompilerInput input : inputs) {
         // Forward-declare all the provided types, so that they
         // are not flagged even if they are dropped from the process.
```

图3 代码源文件示例

3.4 特征提取

相似性特征是研究两个句子之间的语义相关性的一种方法. 如图4所示,本文在比较层中使用了四个相似度比较函数,用以获取错误代码和补丁代码之间的静态特征.

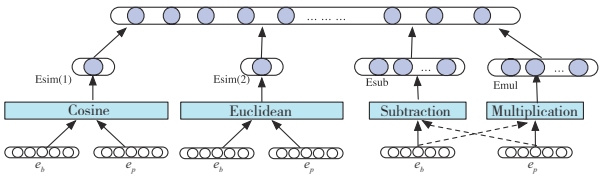


图4　代码静态特征提取过程

4　实验结果与分析

为了评估提出方法的有效性,本节进行了大量的实验探究及结果分析. 实验中模型的初始学习率为1×10^{-5},Batch size为64. 编码器为两个Bi-LSTM层,前一层的输出作为隐藏参数输入到下一层. 第一个Bi-LSTM层的神经元为128,第二个Bi-LSTM的神经元为32. 每个Bi-LSTM层接一个Dropout操作,丢弃率(Dropout rate)为0.5. 单代码片段嵌入向量维度为100. 分类器使用Sklearn包实现. 其中,逻辑回归算法的最大迭代次数为1 000,优化算法选择拟牛顿法(利用损失函数二阶导数矩阵即海森矩阵来迭代优化损失函数),停止求解标准为0.2. 选择线性方法作为SVM的核函数. 决策树和朴素贝叶斯均使用默认参数值. 为避免随机误差,所有实验结果均采用十折交叉验证.

4.1　实验对象

为了训练Doc2vec[12],使用了如表1所示的5个基准集. InferSent[7]则采用RepairThemAll数据集[9]对其进行微调. 实验使用的验证集如表2所示.

表1　Doc2vec训练使用的数据集

Subjects	Correct patches	Incorrect patches	Total
Bears	251	0	251
Bugs.jar	1 158	0	1 158
Defects4J	864	0	864
ManySStubBs4j	34 051	0	34 051
QuixBugs	40	0	40
Total	—	—	36 364

表2　实验使用的数据集

Subjects	Correct patches	Incorrect patches	Total
Liu[11]	137	502	639
Xiong[19]	30	109	139
Defect4j(developers)	356	0	356
Total	523	611	1 134

4.2　评价指标

本文使用了5个指标对实验结果进行评估,分别是准确率Auccuracy、精确率Pecision、召回率Recall、F1值、AUC值. 同时对比模型的ROC曲线,更直观地反映不同模型的预测效果.

4.3　实验设计

为验证融合句嵌入模型和代码特征的补丁验证方法的有效性,本文研究了以下三个问题.

RQ1:不同的句嵌入技术对补丁验证影响?

RQ2:相似性度量特征对补丁验证的影响?

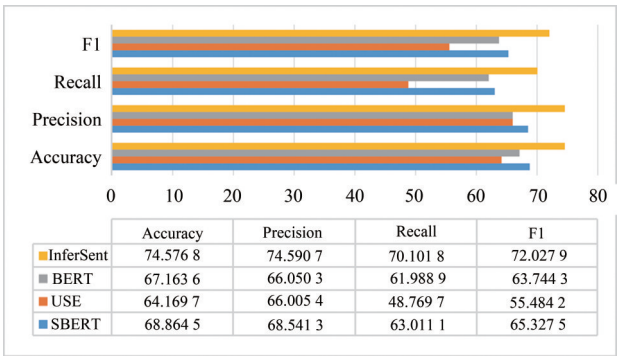
RQ3:与其他补丁验证技术相比,本文提出的模型效果如何?

4.4　实验结果与分析

4.4.1　与不同句嵌入技术比较

为回答问题1,本文使用了其他三种有代表性的句嵌入模型与提出的模型进行了预测能力评估以及泛化能力的对比评估.

图5展示了四种不同句嵌入技术在补丁验证中的效果. 我们从F1分数角度对句嵌入技术进行评估. 一般来说对于机器学习预测模型,F1值越高越好,F1值大于0.7则模型能够较好地预测[13]. 实验结果表明句嵌入模型应用在补丁验证领域极有研究意义,同时本文提出的预测模型具有较好的预测效果.



	Accuracy	Precision	Recall	F1
InferSent	74.576 8	74.590 7	70.101 8	72.027 9
BERT	67.163 6	66.050 3	61.988 9	63.744 3
USE	64.169 7	66.005 4	48.769 7	55.484 2
SBERT	68.864 5	68.541 3	63.011 1	65.327 5

图5　不同句嵌入模型对比柱状图

如图6所示为四个模型进行十折交叉验证的ROC对比图,可看出本文模型的ROC曲线弧度明显大于其余ROC曲线,AUC平均值达到0.82,而其他句嵌入模型的AUC平均值在0.7~0.75之间.

综上,可以回答问题1,不同的句嵌入对补丁验证效果影响很大,从模型综合表现和泛化能力看,InferSent句嵌入模型在补丁验证中的表现都明显优于其他句嵌入模型.

4.4.2　相似性度量特征消融比较

为了回答问题2,本节使用消融实验来验证相似性特征在基于句嵌入技术的静态补丁验证框架下的效果,实验包含六种句嵌入模型BERT[14]、USE[15]、SBERT[16]、InferSent[7]、Doc2vec[12]、InferCode[17]和四种分类器组成的静态补丁验证模型. 其中Doc2vec[12]、InferCode[17]支持向量机线性分类器无法分类. 为保证实验结果可信,其余句嵌入模型采用的交叉特征提取方

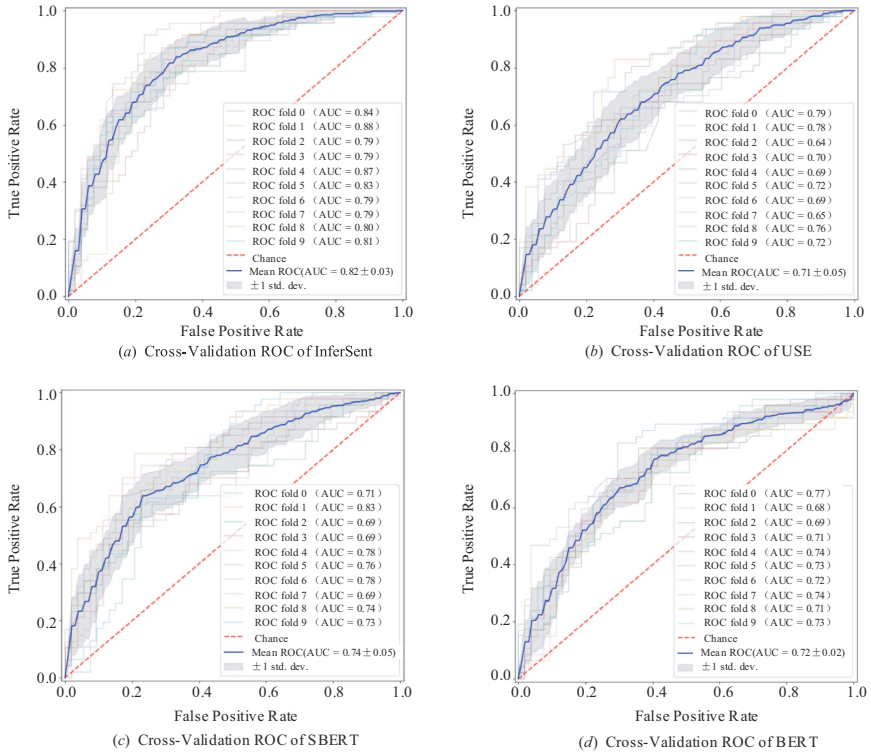


图6 不同句嵌入模型ROC对比图

式与本文方法完全相同.

表3中f1-simi和AUC-simi两列为包含相似性特征的实验结果,f1-nosimi和AUC-nosimi两列为去除相似性特征的实验结果,加粗数据表示包含相似性特征的模型的实验表现优于无相似性特征的模型. 经统计包含相似性特征的模型F1值在17/22(77.3%)的模型中高于不包含相似性的模型,AUC值在16/22(72.7%)的模型中高于不包含相似性的模型. 综上,可以回答问题2,相似度特征可以有效地帮助识别出正确代码片段,对补丁预测有指导意义. 包含相似性特征的预测效果好于不包含相似性特征的补丁验证技术的预测效果.

4.4.3 与其他补丁验证技术比较

RQ3: 与其他补丁验证技术相比,本文提出的模型验证效果如何?

如表4第一行所示是Huang等人[18]提出的结合Doc2Vec和BERT嵌入技术的补丁验证方法,第二行为,Tian等人[10]提出BERT模型结合逻辑回归分类器是最优的补丁验证模型. 可以看出本文的方法的F1值对比两种方法分别提高了11.6%和6.4%.

综合上述对比分析可以回答问题3,与相关的补丁验证技术相比,本文提出的方法在补丁正确性预测上表现出了非常好的性能. 比最新的相关方法在F1值和AUC值上分别提高了11.6%和9%.

表3 相似性度量特征影响统计表

	分类器	F1-simi/%	F1-nosimi/%	AUC-simi	AUC-nosimi
SBERT	朴素贝叶斯	**63.727 0**	**63.426 2**	**0.690 694**	**0.689 350**
USE		64.067 8	64.131 1	0.681 160	0.692 747
BERT		**62.797 6**	**62.768 7**	0.570 620	0.571 320
InferSent		**65.168 6**	**64.962 6**	**0.654 957**	**0.649 451**
doc2vec		**62.329 3**	**61.774 7**	**0.485 267**	**0.480 759**
InferCode		**57.832 6**	**56.721 9**	**0.553 984**	**0.544 641**
SBERT	逻辑回归	**68.783 3**	**66.911 5**	**0.772 642**	**0.765 678**
USE		**51.520 0**	**50.580 1**	**0.701 667**	**0.707 640**
BERT		**67.080 0**	**65.772 9**	**0.750 180**	**0.745 183**
InferSent		**69.715 8**	**69.553 5**	**0.803 635**	**0.799 299**
doc2vec		50.608 8	52.163 9	0.624 414	0.624 450
InferCode		**63.378 0**	**62.813 4**	0.710 765	0.711 709
SBERT	决策树	**57.845 7**	**53.852 8**	**0.611 103**	**0.562 830**
USE		57.546 1	58.596 8	0.596 400	0.600 121
BERT		**59.474 6**	**58.129 4**	**0.619 414**	**0.614 803**
InferSent		**64.020 9**	**61.470 4**	**0.658 836**	**0.636 090**
doc2vec		**51.162 0**	**48.544 5**	**0.539 753**	**0.518 362**
InferCode		**57.165 4**	**55.247 2**	**0.592 403**	**0.585 357**
SBERT	支持向量机	**65.327 5**	**64.847 6**	0.733 010	0.736 453
USE		55.484 2	60.775 1	**0.762 580**	**0.715 587**
BERT		**63.744 3**	**42.724 7**	**0.724 932**	**0.619 301**
InferSent		72.027 9	74.157 3	**0.834 203**	**0.814 131**

表4　补丁验证技术的验证效果比较

模型	准确率	精确率	召回率	F1值
Doc2vec+BERT	0.664 2	0.653 5	0.558 9	0.602 5
BERT+LR	0.685 6	0.668 8	0.645 2	0.654 6
本模型	0.735 1	0.736 3	0.702 3	0.718 9

4　结论

本文提出了一种新的补丁验证方法,其可以提高补丁验证正确率并降低补丁验证开销,同时研究了句嵌入模型结合不同分类算法对补丁验证效果的影响.实验表明,本文方法可以识别70%以上的正确补丁,能够有效预测补丁正确性,并具有良好的泛化能力.同时,实验证明了相似度特征可以有效地帮助识别出正确代码片段,包含相似性特征的补丁预测模型的F1值和AUC值分别在77.3%和72.7%的情况下高于不包含相似性特征的补丁预测模型.

参考文献

[1] GAZZOLA L, MICUCCI D, MARIANI L. Automatic software repair: A survey[J]. IEEE Transactions on Software Engineering, 2017, 45(1): 34-67.

[2] LE X B D, BAO L, LO D, et al. On reliability of patch correctness assessment[C]//2019 IEEE/ACM 41st International Conference on Software Engineering. Montreal: IEEE, 2019: 524-535.

[3] WANG S, WEN M, LIN B, et al. Automated patch correctness assessment: How far are we?[C]//35th IEEE/ACM International Conference on Automated Software Engineering. Melbourne: IEEE, 2020: 968-980.

[4] HINDLE A, BARR E T, SU Z, et al. On the naturalness of software[C]//34th International Conference on Software Engineering (ICSE). Zurich: IEEE, 2012: 837-847.

[5] CSUVIK V, HORVÁTH D, HORVÁTH F, et al. Utilizing source code embeddings to identify correct patches[C]//2nd International Workshop on Intelligent Bug Fixing (IBF). London: IEEE, 2020: 18-25.

[6] 齐玉华. 软件自动修复关键技术研究[D]. 长沙: 国防科学技术大学, 2013.
QI Y H. Research on Key Technologies of Software Automatic Repair[D]. Changsha: National University of Defense Technology, 2013. (in Chinese)

[7] CONNEAU A, KIELA D, SCHWENK H, et al. Supervised learning of universal sentence representations from natural language inference data[C]//2017 Conference on Empirical Methods in Natural Language Processing. Copenhagen: ACL, 2017: 670-680.

[8] BOWMAN S R, ANGELI G, POTTS C, et al. A large annotated corpus for learning natural language inference[C]//Proceedings of the Conference on Empirical Methods in Natural Language Processing. Lisbon: ACL, 2015: 632-642.

[9] DURIEUX T, MADEIRAL F, MARTINEZ M, et al. Empirical review of Java program repair tools: A large-scale experiment on 2141 bugs and 23551 repair attempts[C]//European Software Engineering Conference and Symposium on the Foundations of Software Engineering. Tallinn: ACM, 2019: 302-313.

[10] TIAN H, LIU K, KABORÉ A K, et al. Evaluating representation learning of code changes for predicting patch correctness in program repair[C]//35th IEEE/ACM International Conference on Automated Software Engineering. Melbourne: IEEE, 2020: 981-992.

[11] LIU K, WANG S, KOYUNCU A, et al. On the efficiency of test suite based program repair: A systematic assessment of 16 automated repair systems for java programs [C]//42nd International Conference on Software Engineering. Seoul: ACM, 2020: 615-627.

[12] LE Q, MIKOLOV T. Distributed representations of sentences and documents[C]//31st International Conference on Machine Learning. New York: ACM, 2014: 1188-1196.

[13] DAVIS J, GOADRICH M. The relationship between precision-recall and ROC curves[C]//23rd International Conference on Machine Learning. New York: ACM, 2006: 233-240.

[14] DEVLIN J, CHANG M W, LEE K, et al. BERT: Pre-training of deep bidirectional transformers for language understanding[C]//Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Minneapolis: ACL, 2019: 4171-4186.

[15] CER D, YANG Y, KONG S, et al. Universal sentence encoder for English[C]//Conference on Empirical Methods in Natural Language Processing. Brussels: ACL, 2018: 169-174.

[16] REIMERS N, GUREVYCH I. Sentence-bert: Sentence embeddings using siamese bert-networks[C]//Conference on Empirical Methods in Natural Language Processing. Hong Kong: ACL, 2019: 3980-3990.

[17] BUI N D Q, YU Y, JIANG L. Infercode: Self-supervised learning of code representations by predicting subtrees [C]//43rd International Conference on Software Engineering. Madrid: IEEE, 2021: 1186-1197.

[18] 黄颖, 姜淑娟, 蒋婷婷. 结合Doc2Vec和BERT嵌入技术的补丁验证方法[J]. 计算机科学, 2022, 49(11): 83-89.
HUANG Y, JIANG S J, JIANG T T. Patch validation approach combining Doc2Vec and BERT embedding technologies[J]. Computer Science, 2022, 49(11): 83-89. (in Chinese)

[19] XIONG Y, LIU X, ZENG M, et al. Identifying patch correctness in test-based program repair[C]//International Conference on Software Engineering. New York: ACM, 2018: 789-799.

作者简介

蒋婷婷 女, 1997年10月出生于安徽淮北. 目前就读于中国矿业大学. 主要研究方向为软件分析与测试、程序缺陷自动修复.
E-mail: TS20170072P31@cumt.edu.cn

姜淑娟(通讯作者) 女, 1966年12月出生于山东莱阳. 现为中国矿业大学教授、博士生导师. 主要研究方向为软件分析与测试、缺陷预测、故障定位、进化计算.
E-mail: shjjiang@cumt.edu.cn