doi:10.11920/xnmdzk.2025.03.008

强化文本长程依赖和多特征融合的重复软件缺陷报告检测方法

谢　琪[1,2],刘彦辰[1]

(1. 西南民族大学计算机与人工智能学院,四川 成都 610041;
2. 西南民族大学计算机系统国家民委重点实验室,四川 成都 610041)

摘　要:自动重复软件缺陷检测是缺陷处理流程中的关键环节,其性能直接影响缺陷解决的整体效率.针对现阶段研究中预训练模型的长度限制问题和特征单一问题,提出一种强化文本长程依赖和多特征融合的重复软件缺陷报告检测方法.该方法通过结合 BiLSTM-Attention 机制与 Longformer 预训练模型,增强对篇章级文本的长程语义信息捕捉能力,从而提取更准确的语义相似度特征.其次,针对结构化元数据类别信息,构建特征提取网络以抽取类别相似度特征.最终,将上述特征与词组重叠特征融合后训练分类模型,以实现高效的重复缺陷报告检测.通过在 Bugzilla、JIRA 和 GitHub 平台的 Eclipse、NetBeans、OpenOffice、Hadoop 和 VSCode 项目上进行实验,结果表明,与基线方法相比,所提方法在 F1 分数和 Accuracy 上分别平均提升了 2.40%和 2.12%,在跨平台场景下实现了更优的检测性能.

关键词:重复软件缺陷报告检测;Longformer;语义相似度;长程依赖

中图分类号:TP311　　**文献标志码**:A　　**文章编号**:2095-4271(2025)03-0298-10

A duplicate software defect report detection method enhanced with long-range text dependency modeling and multi-feature fusion

XIE Qi[1,2],LIU Yanchen[1]

(1. School of Computer Science and Artificial Intelligence,Southwest Minzu University,Chengdu 610041,China;
2. State Ethnic Affairs Commission Key Laboratory for Computer Systems,Southwest Minzu University,Chengdu 610041,China)

Abstract:Automatic duplicate software defect detection is a critical component of the defect handling process,as its performance directly impacts the overall efficiency of defect resolution.Addressing the limitations of pre-trained models in sequence length and feature diversity,this paper proposed a duplicate bug report detection method enhanced with long-range text dependency modeling and multi-feature fusion.The proposed method combined BiLSTM-Attention and the Longformer pre-trained model to improve the ability to capture long-range semantic information in paragraph-level texts,thereby extracting more precise semantic similarity features. Furthermore,a feature extraction network was constructed to capture category similarity features from structured metadata.Finally,these extracted features were fused with token overlap features and used to train a classification model for efficient duplicate defect report detection.Experiments were conducted on large-scale software defect report datasets from Bugzilla,JIRA,and GitHub platforms,specifically the Eclipse,NetBeans,OpenOffice,Hadoop,and VSCode projects. The results demonstrated that,compared to baseline methods,the proposed approach achieved an average improvement of 2.40% in F1 score and 2.12% in Accuracy,confirming its superior detection performance in cross-platform scenarios.

Keywords:duplicate bug report detection;Longformer;semantic similarity;long range dependency

收稿日期:2025-03-05
作者简介:谢琪(1983-),女,副教授,博士,研究方向:服务计算、深度学习.E-mail:qi.xie.swun@gmail.com
通信作者:刘彦辰(2000-),男,研究方向:日志重复性检测.E-mail:swun_lyc@163.com
基金项目:四川省科技计划项目(25GJHZ0156)

随着软件系统的持续扩展,软件缺陷管理已成为开发与维护过程中的关键挑战[1].用户通过提交缺陷报告(BR)至 Bugzilla 等追踪系统实现缺陷管理[2],报告经分类后分配至相关人员处理.然而,跨项目信息共享机制的缺失导致缺陷追踪系统中存在大量重复报告,造成严重的资源浪费.作为缺陷管理流程的初始环节,自动重复缺陷报告检测(DBRD,Duplicate Bug Report Detection)通过识别重复报告降低人力成本,并为同类缺陷提供信息补充,因此成为软件工程领域的重要研究方向.

DBRD 的研究主体是软件缺陷报告,单份完整的软件缺陷报告通常由结构化的元数据和非结构化文本组成,前者通过枚举值限定缺陷的类别,包括创建时间、优先级等字段,后者以自然语言的形式对缺陷进行更加细致的叙述,主要由标题和描述字段构成.早期的 DBRD 研究主要基于信息检索技术展开,这类方法主要通过结构化属性检测重复报告,但无法有效利用非结构化属性中的文本语义信息,因此效果较差.为此,现阶段研究中通常会引入包括卷积神经网络 TextCNN、序列网络 RNN、长短期记忆网络 LSTM 以及预训练模型 BERT 等深度学习技术用于衡量非结构化属性的文本相似性.其中 BERT 类模型性能优越,但仍存在局限性:一方面,预训练模型存在输入长度限制(通常≤512 token),在处理非结构化属性文本时存在信息遗失问题;另一方面,现有方法特征组成较为单一,未能有效利用优先级、组件关联性等结构化元数据.

针对以上问题,本文提出一种强化文本长程依赖和多特征融合的重复软件缺陷报告检测方法.该方法通过结合 Longformer 中稀疏注意力机制和 BiLSTM-Attention 实现缺陷报告中篇章级文本的长程依赖建模,克服传统动态词嵌入模型的输入序列长度限制,从而生成完整文本的语义相似度特征;其次,采用传统的信息检索方式进行文本词符比对,生成词组重叠度特征;同时针对缺陷报告的元数据信息构建特征提取网络提取类别相似度特征.最终,将三类特征融合后通过分类模型实现缺陷报告对的重复性检测.

1　相关研究

1.1　基于信息检索技术的相关工作研究

早期研究主要采用信息检索(IR)技术进行重复缺陷报告检测.Runeson 等人[3]使用词袋模型(BoW)结合余弦相似度度量文本相似性.Wang 等人[4]在此基础上引入词频加权机制,并将堆栈跟踪分析融入标题与描述的相似度计算.Sun 等人[5]则采用 TF-IDF 特征结合 SVM 分类器进行判别.BM25F 是一种短文本相似度计算函数,Aggarwal 等人[6]基于该模型提出一种建立上下文特征的检测方法.Sun 等人[7]对 BM25F 函数进行较长文本的适用性扩展,提出 REP 方法进一步优化了检索效率.Nguyen 等人[8]发现 BM25F 在处理跨术语描述场景时存在局限,因此提出融合潜在狄利克雷分布(LDA)的 DBTM 模型以提高准确性.

传统 IR 方法虽能捕捉词汇频率特征,却无法处理上下文语义关联.为此,Sureka 等人[9]基于 n-gram 模型开发字符级检测方法,通过语法结构分析增强语义理解.Sabor 等人[10]进一步扩展该方法,结合堆栈跟踪中的函数调用序列特征提升检测能力.Wang 等人[11]提出多模态检测框架,融合文本相似度(TF-IDF+词嵌入)与截图特征(结构+颜色属性)生成候选列表.

1.2　基于深度学习的相关工作研究

深度学习技术推动了 DBRD 研究的革新.Deshmukh 等人[12]构建暹罗网络架构,联合 TextCNN(提取全局特征)与 LSTM(建模局部时序依赖)实现语义匹配.Akilan 等人[13]通过双层 LDA 主题建模加速检索过程.Xie 等人[14]提出基于词嵌入与深度网络的分布式表示方法,并结合结构化属性进行分类决策.周文杰等人[15]在 GRU 网络中引入注意力机制以动态计算文本片段相似度,辅以 LDA 主题特征与类别相似度特征提升性能.Lazar 等人[16]提出 D_TS 方法,通过 25 种结构化相似度特征优化分类效果,曾杰等人[17]在此基础上融合 Doc2Vec 文本特征实现性能的进一步提升.

随着 Transformer 架构的普及,基于 BERT 的解决方案成为主流.曾方等人[18]提出 D_BBAS 方法,引入 BERT 模型计算短文本字段相似度,使用 Doc2Vec 模型计算长文本字段相似度,随后将两种相似度特征与 D_TS 方法中 25 种结构化相似度特征融合后共同检测重复项.Messaoud 等人[19]提出 BERT-MLP 方法,截取符合长度限制的前部文本作为 BERT 模型的输入,多层感知机分类器(MLP)生成检测结果.Isotani 等人[20]针对文本交互方案下 BERT 模型检测效率不足

的问题,引入基于 Siamese 结构的 Sentence-BERT 模型提取文本语义特征.

2 本文方法

DBRD(Duplicate Bug Report Detection)的目标是准确检测当前缺陷报告与已有缺陷报告间的重复性,通常分为特征提取和重复性分类两个阶段.特征提取阶段针对缺陷报告的不同字段,提取可离线存储的向量表示,以完成相似度特征计算.重复性分类阶段利用多种相似度特征训练的分类模型返回缺陷报告对的重复性检测结果.

为达成以上目标,本文以当前主流缺陷管理系统 Bugzilla、JIRA 和 GitHub 的缺陷报告集合为研究对象,提出一种强化文本长程依赖和多特征融合的重复软件缺陷报告检测方法,如图 1 所示,主要包含五个核心模块:数据预处理、文本语义相似度特征提取、元数据相似度特征提取、词组重叠度计算和分类模型.数据预处理的目的是过滤缺少信息的无效报告(例如标题和描述内容皆为空的报告对),同时将非结构化属性中的标题和描述字段链接为完整文本;文本语义相似度特征对链接后的文本进行动态的语义编码,利用余弦相似度返回语义相似度特征;元数据相似度特征提取模块针对结构化属性中具有固定格式的元数据构建相似度特征提取网络,获取元数据相似度;词符重叠度计算从文本比对角度获取文本中词组的相似度;分类模型以三种相似度特征为输入,返回最终的判定结果.

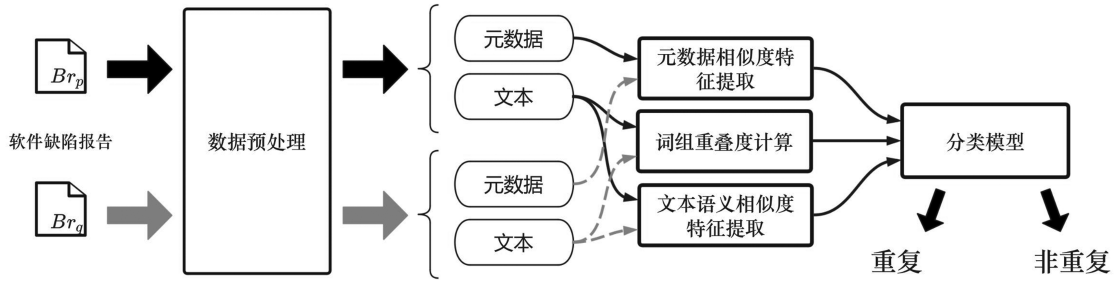

图 1 方法总体架构

Fig. 1 General framework of the method

2.1 文本语义特征提取

非结构化属性体现为缺陷报告中以自然语言描述的缺陷细节(如错误现象、复现步骤等),通常篇幅较长,传统信息检索方法难以有效捕捉上下文语义关联,而 BERT 等基于 Transformer 架构的预训练模型虽然表征能力强大,但由于预训练阶段仅考虑了短文本场景且采用绝对位置编码,无法有效处理长序列文本.现阶段处理长文本时的主流方式是先将文本划分为多个片段[21],利用编码器分别针对片段进行语义嵌入,最终利用语义聚合将各个编码器输出的嵌入表示归纳为固定维度的综合语义表示.这种方法能够捕捉到片段内的语义联系,但片段之间难以建立足够的关联,导致语义表示的准确性不足.针对这一问题,本模块结合 Longformer 和 BiLSTM Attention 抽取非结构化属性文本的相似度.模型结构如图 2 所示,主要构成为:嵌入层、语义归纳层、PCA 层和余弦相似度计算,接下来将从这四个层面进行阐述.

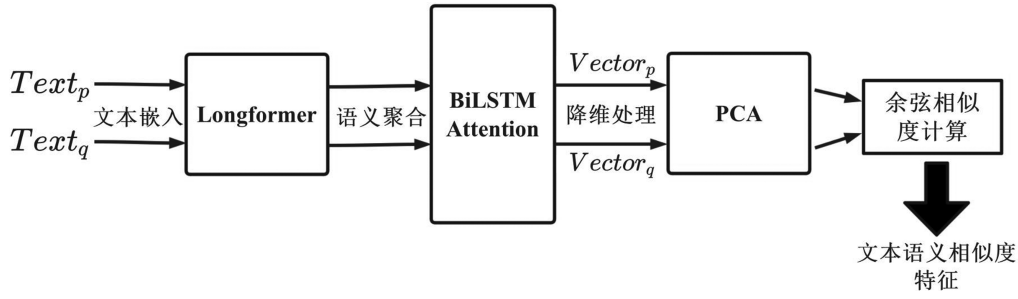

图 2 语义相似度特征提取

Fig. 2 Semantic similarity feature extraction

1) 嵌入层

嵌入层的目的是将非结构化的自然语言文本转换为高维稠密向量表示,以便后续模型能够有效捕捉语义信息.鉴于缺陷报告的文本长度通常达到篇章级别,本研究采用预训练模型 Longformer 进行文本词嵌入.Longformer 由 Beltagy 等人[22]提出,是一种专为长文本处理设计的预训练语言模型.该模型采用与 BERT 类似的 12 层 Transformer 编码器结构,并在其基础上扩展了位置编码长度,同时引入稀疏注意力机制,以降低计算复杂度,使其能够在大规模长文本数据集上进行高效预训练.借助这一机制,Longformer 能够直接建模长序列文本中的语义依赖关系,并通过少量微调即可在下游任务中实现优越的性能.

Longformer 的稀疏注意力核心机制包括滑动窗口注意力(Sliding Window Attention)和全局注意力(Global Attention)(见图 3):前者使用固定长度 w 的滑动窗口,使每个 token 仅与其前后共 $w-1$ 个 token 进行交互,从而捕捉局部语义信息;后者根据任务需求,设定关键 token(如[CLS]标记)与其余所有 token 进行交互,以捕捉长程的双向依赖关系.由于 Global Attention 依赖于关键 token 位置的选择,而在重复性检测任务中,token 的重要性是动态变化的,因此本文中仅采用滑动窗口注意力机制对文本进行初步嵌入.针对任意缺陷报告 q,Longformer 编码后的词向量表示序列如下:

$$E^q=\left[e^q_{CLS},e^q_{w_1},e^q_{w_2},\ldots,e^q_{w_i},\ldots,e^q_{w_N},e^q_{SEP}\right]\in\mathbb{R}^{(N+2)\times d}. \quad (1)$$

式中:N 为非结构化属性文本中描述和标题字段拼接后的总长度;e_{CLS} 和 e_{SEP} 分别为文本开头和结尾的特殊标记;$e^q_{w_i}\in\mathbb{R}^{1\times d}$ 为第 i 个 token 的 d 维上下文感知向量,其感受野覆盖 $\left[i-\frac{w-1}{2},i+\frac{w-1}{2}\right]$ 区间内的上下文信息.

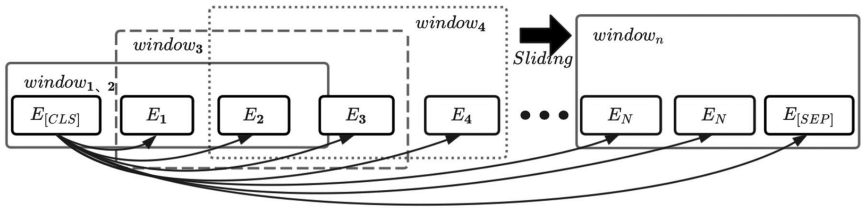

图 3　滑窗注意力机制和全局注意力机制

Fig. 3　Sliding Window Attention and Global Attention mechanism

2) 语义归纳层

Sliding Window Attention 机制下的注意力窗口通常具备堆叠部分,随着 Transformer 层数的增加,头部序列的语义信息能够逐层向后传播.这种方式能够有效建立较为连续的语义依赖,但是依旧受到窗口长度的限制.为此,该层结合基于 Siamese 结构的 BiLSTM 网络和 Attention 机制,将 E^q 中分布的词向量表示逐步归纳为固定维度的全局语义表示.

BiLSTM 网络的作用是通过单元记忆和门控机制控制信息双向流动,从而捕捉长距离依赖关系.对于词向量序列 E^q,BiLSTM 逐步计算序列首端至尾端时间步 t 的门控状态.在此过程中,网络首先根据 t 时刻的嵌入表示 $e^q_{w_t}$ 计算遗忘门 f_t,决定前序信息的遗忘和保留,表达式为:

$$f_t=\sigma\left(W_f\left(\vec{h}_{t-1}\oplus e^q_{w_t}\right)+b_f\right). \quad (2)$$

输入门 i_t 筛选当前时刻词表示 $e^q_{w_t}$ 的有效信息,生成候选记忆 $\tilde{C}_t$.随后协同遗忘门 f_t 和前序记忆单元 $\vec{C}_{t-1}$ 共同计算得出当前时刻的记忆单元 $\vec{C}_t$,表达式为:

$$f_t=\sigma\left(W_i\left(\vec{h}_{t-1}\oplus e^q_{w_t}\right)+b_i\right). \quad (3)$$

$$\tilde{C}_t=tanh\left(W_C\left(\vec{h}_{t-1}\oplus e^q_{w_t}\right)+b_C\right). \quad (4)$$

$$\vec{C}_t=f_t\odot\vec{C}_{t-1}+i_t\odot\tilde{C}_t. \quad (5)$$

输出门 o_t 决定 t 时刻隐态 $\vec{h}_t$ 保留 $\vec{C}_t$ 信息的比例,该过程计算公式如下:

$$o_t=\sigma\left(W_o\left(\vec{h}_{t-1}\oplus e^q_{w_t}\right)+b_o\right). \quad (6)$$

$$\vec{h}_t=o_t\odot tanh\vec{C}_t. \quad (7)$$

以上过程可逐步生成 E^q 对应的前向时间步隐态

序列 $\vec{h}=\left\{\vec{h}_1,\vec{h}_2,...,\vec{h}_{N+2}\right\}$,为同时捕捉 E^q 的双向依赖关系,接下来从序列尾端至首端重复以上过程,得到反向时间步隐态序列 $\overleftarrow{h}=\{\overleftarrow{h}_1,\overleftarrow{h}_2,...,\overleftarrow{h}_{N+2}\}$. $\vec{h}_t$ 与 $\overleftarrow{h}_t$ 水平拼接后得到双向隐态序列 $\overleftrightarrow{h}_t$,表达式如下:

$$\overleftrightarrow{h}_t=\left\{\vec{h}_1\oplus\overleftarrow{h}_1,\vec{h}_2\oplus\overleftarrow{h}_2,...,\vec{h}_{N+2}\oplus\overleftarrow{h}_{N+2}\right\}=\left\{\vec{h}_1,\vec{h}_2,...,\vec{h}_{N+2}\right\}\in\mathbb{R}^{(N+2)\times 2d}. \tag{8}$$

引入注意力机制的目的是根据不同时间步的重要程度为 $\overleftrightarrow{h}_t$ 分配全局关联权重,防止信息传递过程中遗失远距离时间步的依赖关系,同时达到该层语义信息聚合的目的:

$$\alpha_t=softmax\left(\frac{\overleftrightarrow{h}_tW_Q\left(\vec{h}_{N+2}W_k\right)^T}{\sqrt{d_k}}\right). \tag{9}$$

$$\hat{g}=\sum_{t=1}^{N+2}\alpha_t\overleftrightarrow{h}_t\in\mathbb{R}^{1\times 2d}. \tag{10}$$

式中:W_Q,W_k 为可训练的参数矩阵;$\hat{g}$ 为最终聚合后的综合语义表示.

3) PCA 层

聚合后的语义向量通常会被离线存储作为缺陷报告非结构化属性的特征表示,为减少空间损耗,该层采用主成分分析(PCA)压缩高维语义表征 $\hat{g}$ 的冗余信息,实现特征解耦,其优化目标为:

$$W^*=argmax\ tr\left(W^T\Sigma W\right). \tag{11}$$

式中:$\Sigma=\frac{1}{M}\sum_{i=1}^{M}\left(\hat{g}_i-\mu\right)\left(\hat{g}_i-\mu\right)^T$ 为协方差矩阵;M 为训练样本数;μ 为均值向量.

随后通过特征值分解选取前 k 个最大特征值对应正交基向量构建投影矩阵 $W\in\mathbb{R}^{2d\times k}$,将原始 $2d$ 维向量 $\hat{g}_i$ 映射至低维空间,从而在保持准确性的同时减少空间损耗:

$$\hat{e}_i=W^T\left(\hat{g}_i-\mu\right)\in\mathbb{R}^k. \tag{12}$$

4)余弦相似度计算

余弦相似度是一种衡量两个向量在高维空间中夹角相似度的度量方法,因为对向量大小不敏感和计算简单而常用于文本相似度计算任务中.对于任意缺陷报告对降维后的语义向量 $\hat{e}_p$ 和 $\hat{e}_q$,余弦相似度计算公式为:

$$feature_1=Cosine\ Similarity\left(\hat{e}_p,\hat{e}_q\right)=\frac{\sum_{i=1}^{d}\hat{e}_{p_i}\times\hat{e}_{q_i}}{\sqrt{\sum_{i=1}^{d}\left(\hat{e}_{p_i}\right)^2}\times\sqrt{\sum_{i=1}^{d}\left(\hat{e}_{q_i}\right)^2}}. \tag{13}$$

式中,*Cosine Similarity* 为余弦相似度,数值越接近 1 表示 $\hat{e}_p$ 和 $\hat{e}_q$ 越相似,越接近 0 表示越不相关.

2.2 元数据类别特征提取

软件缺陷报告的结构化属性包括产品、组件、版本、创建日期和缺陷编号五种元数据字段,其中产品和组件通常为类别数量固定的枚举字段,版本、创建日期和缺陷编号则为浮点类型或整型数据.对于由若干词符构成的产品和组件字段,二者综合后可参与重复性判别特征.该模块将二者水平拼接后采用 one-hot 编码方式映射为多维稀疏向量,随后构建浅层 Siamese 网络学习差异特征,最终通过余弦相似度计算得出重复性判别特征,网络结构表达式为:

$$y_q=Dropout\left(ReLU\left(W_qx_{product}\oplus x_{component}+b_q\right)\right)\in\mathbb{R}^{1\times k}. \tag{14}$$

$$y_q=Dropout\left(ReLU\left(W_qx_{product}\oplus x_{component}+b_q\right)\right)\in\mathbb{R}^{1\times k}. \tag{15}$$

$$feature_2=Cosine\ Similarity\left(y_p,y_q\right). \tag{16}$$

式中:*Dropout* 代表正则化;*ReLU* 为激活函数;$W_q\in\mathbb{R}^{32\times k}$ 为可训练的参数矩阵.

对于数值类型的版本、创建日期和缺陷编号字段,仅需分别衡量报告对间三种属性的数值差异即可,计算方式如下:

$$feature_i=\frac{1}{\exp\left(br_i^q-br_i^p\right)}. \tag{17}$$

式中:br_i^q 为缺陷报告 q 的 $i\in[1,3]$ 号特征,$feature_1$、$feature_2$、$feature_3$ 分别为版本、日期和编号字段的相似度特征;exp 为绝对值函数.

2.3 词符重叠特征计算

软件缺陷报告通常为描述技术问题的专业性文档,其中涵盖大量的专业领域词汇.语义相似度特征能够从语义角度反映文本综合信息的相似程度,词符重叠率则能够从文本比对的角度为相似度判定作出贡献.为量化缺陷报告 p 和缺陷报告 q 文本中词符序列 S_p 和 S_q 的重叠度,本模块引入 Saric 等人[23] 在 Simple

TakeLab 系统中的重叠度特征参与重复性检测.

1) 文本词形还原后的重叠率:

$$feature_6 = 2\left(\frac{|S_p|}{|S_p \cap S_q|}+\frac{|S_q|}{|S_p \cap S_q|}\right)^{-1}. \tag{18}$$

2) 基于 WordNet 增强的单词重叠率(WordNet-augmented word overlap):

$$feature_7 = 2\left(\frac{|S_p|}{\sum_{w_q \in S_q} score(w_q, S_p)}+\frac{|S_p|}{\sum_{w_p \in S_p} score(w_p, S_q)}\right)^{-1}. \tag{19}$$

$$score(w,S)=\begin{cases}1, w \in S\\ max\, sim(w,w'), else\end{cases}. \tag{20}$$

式中:*sim* 计算 WordNet 网络中词汇 w 和 w' 最短路径长度的倒数.

3)加权词汇重叠率(Weighted Word Overlap):

$$feature_8 = 2\left(\frac{\sum_{w' \in S_p} i(w')}{\sum_{w \in S_p \cap s_q} i(w)}+\frac{\sum_{w' \in S_q} i(w')}{\sum_{w \in S_p \cap s_q} i(w)}\right)^{-1}. \tag{21}$$

$$i(w)=\ln\frac{\sum_{w' \in C} freq(w')}{freq(w)}. \tag{22}$$

式中:*freq* 函数用于计算词频;C 为缺陷报告的语料库.

2.4　分类模型

该模块通过将从结构化属性和非结构化属性中提取的相似度特征集($feature_1 \sim feature_8$)拼接后输入分类模型,以完成训练和最终的重复性检测任务.为了降低单一分类算法可能引入的偏差,并确保模型的鲁棒性和泛化能力,本文采用 K 近邻(KNN)、随机森林(Random Forest)、线性支持向量机(Linear SVM)以及多层感知机(MLP)四种经典分类算法对所提出的方法进行评估.

3　实验设计与结果分析

3.1　数据集说明

实验数据集来源于三种主流缺陷管理系统中的开源软件真实缺陷报告集,具体包括 Lazar 等人[16]在 Bugzilla 平台上采集的 Eclipse、NetBeans 和 OpenOffice 缺陷报告,Zhang 等人[24]收集的 JIRA 平台上的 Hadoop 缺陷报告,以及 GitHub 平台的 VSCode 缺陷报告集.这五个数据集均以 $(br_1, br_2, isduplicate)$ 的三元组形式存储.首先,对原始数据中的重复三元组进行去重处理,如(38476,1528,1)和(1528,38476,1)等.然后,在相同的数据划分和分类模型参数设置下与对照方法进行对比实验.去重后五种数据集的规模以及划分如表 1 所示.

表 1　数据规模及划分

Table 1　Dataset scale partition

平台	数据集	报告对总数(份)	训练集		测试集	
			正样本(份)	负样本(份)	正样本(份)	负样本(份)
Bugzilla	Eclipse	247 362	3 000	6 000	85 192	153 170
	NetBeans	185 581	3 000	6 000	92 827	83 754
	OpenOffice	99 086	3 000	6 000	54 289	35 797
JIRA	Hadoop	6 715	736	736	540	4 703
GitHub	VSCode	30 745	7 126	10 899	3 712	9 008

3.2　评价指标

本文选取重复软件缺陷报告检测任务中常用的准确性(Accuracy)、精确度(Precision)、召回率(Recall)和 F1 分数作为评价指标,四种指标的计算公式如下:

$$Accuracy=\frac{TP+TN}{TP+TN+FP+FN}. \tag{23}$$

$$Precision=\frac{TP}{TP+FP}. \tag{24}$$

$$Recall=\frac{TP}{TP+FN}. \tag{25}$$

$$F1=2\left(Recall^{-1}+Precision^{-1}\right)^{-1}. \tag{26}$$

式中:T(True)代表模型预测正确,F(False)代表模型预测错误,P(Positives)代表正类样本,即重复报告,N(Negatives)代表负类样本,即非重复报告;TP,TN 分别代表正确预测的正、负样本数目,FP,FN 分别代表错误预测为正、负样本数目.

3.3　对比实验

为验证本文方法有效性,以 BERT-MLP[19]、D_TS[16]和曾杰所提方法[17]为对照方法,采用相同数据

划分和分类模型配置,首先分别针对 Bugzilla 平台的 Eclipse、NetBeans 和 OpenOffice 公开数据集进行方法性能评估,对比结果见表 2 至表 6.

表 2 Eclipse 数据集对比结果

Table 2 Comparison results on Eclipse dataset 单位:百分比(%)

方法	MLP				KNN				Linear SVM				Random Forest			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
BERT-MLP	81.11	79.27	81.33	79.90	78.26	76.88	79.37	77.26	80.43	78.74	81.07	79.33	80.48	78.57	80.45	79.18
D_TS	95.93	96.21	94.81	95.45	94.20	94.65	92.56	93.47	95.83	96.03	94.75	95.34	95.62	96.15	94.22	95.07
文献[17]方法	96.17	96.42	95.14	95.73	94.70	95.12	93.19	94.04	**96.92**	**97.08**	**96.12**	**96.57**	96.28	96.34	95.43	95.86
本文方法	**97.05**	**97.11**	**96.37**	**96.72**	**96.30**	**96.60**	**95.24**	**95.86**	96.80	96.94	95.99	96.44	**96.93**	**97.22**	**96.02**	**96.58**

表 3 NetBeans 数据集对比结果

Table 3 Comparison results on NetBeans dataset 单位:百分比(%)

方法	MLP				KNN				Linear SVM				Random Forest			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
BERT-MLP	78.33	78.31	78.30	78.31	76.62	76.75	76.50	76.53	77.66	77.72	77.58	77.60	78.40	78.39	78.42	78.39
D_TS	90.44	91.25	90.80	90.43	86.94	88.57	87.44	86.88	89.05	90.15	89.47	89.03	91.24	91.90	91.57	91.24
文献[17]方法	93.29	93.56	93.52	93.29	88.04	89.39	88.50	88.00	92.28	92.73	92.56	92.28	92.93	93.35	93.20	92.93
本文方法	**93.85**	**94.65**	**94.68**	**94.49**	**94.49**	**94.04**	**94.05**	**93.85**	**93.68**	**93.96**	**93.91**	**93.68**	**94.60**	**94.73**	**94.78**	**94.59**

表 4 OpenOffice 数据集对比结果

Table 4 Comparison results on OpenOffice dataset 单位:百分比(%)

方法	MLP				KNN				Linear SVM				Random Forest			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
BERT-MLP	83.70	83.16	83.27	83.21	82.41	81.81	82.33	82.01	82.50	81.99	82.79	82.20	81.97	81.79	82.79	81.79
D_TS	91.34	90.69	92.48	91.13	84.17	84.81	86.53	84.05	89.65	89.21	91.15	89.46	88.71	88.54	90.53	88.55
文献[17]方法	94.17	93.46	94.98	93.99	85.58	85.90	87.74	85.44	94.15	93.44	94.97	93.97	91.80	91.10	92.80	91.59
本文方法	**94.57**	**93.89**	**95.15**	**94.38**	**92.77**	**92.09**	**93.88**	**92.58**	**94.19**	**93.49**	**95.08**	**94.02**	**93.18**	**92.51**	**94.29**	**93.01**

以上三组实验中,BERT-MLP 方法的性能表现较差,原因一方面源于该方法仅利用了文本部分的语义特征,特征组成较为单一.另一方面在于抽取的文本信息受限于 BERT 序列长度限制,该方法处理非结构化属性文本时,首先对标题和描述文本进行截断处理,随后输入 BERT 进行语义特征提取,在这个过程中通常会遗失关键的检测信息.相较于 BERT-MLP,本文方法结合 Longformer 和 BiLSTM Attention 增强模型的长程依赖关系捕捉能力,避免了信息遗失的情况.同时,结合了非结构化属性的元数据相似性特征和词符重叠率特征,因此取得了更优的效果.

D_TS 方法主要基于多种信息检索技术,通过对字段相似度特征进行多样性组合检测重复项.该方法提供了 25 种结构化的相似度特征计算方式,采用了更全面的特征提取方式,因此相比于 BERT-MLP,在 Bugzilla 平台数据上取得了更好的效果.然而,该方法中文本语义相似度特征是在 New York Times 和 Wikipedia 语料库上采用潜在语义分析(LSA)得出,无法有效利用文本的双向语义信息,因此相比于本文方法性能较弱.

曾杰所提方法在 D_TS 的基础上,融入了由 Doc2Vec 模型抽取的分布式语义相似度,通过强化非结构化属性文本特征提取,达到了更好的效果.该方法的缺点在于,Doc2Vec 模型对文本进行建模时,仅能捕捉局部且单向的依赖关系,且只能生成静态的文档向量表示,无法解决一词多义情况.在语境复杂的数据集上(如 OpenOffice),存在语义相似度不够准确的问题.而本文方法采用了能够捕捉双向语义的预训练长文本语言模型,并引入注意力机制为每个时间步中的关键信息动态赋予权重,因此能够用较少特征达到更优的检测性能,在 Eclipse、NetBeans 和 OpenOffice 项目上相比于性能最好的基准方法,F1 平均提升 0.85%、2.52%和 2.25%,Accuracy 平均提升 0.76%,2.52%和 2.25%.

接着,分别针对 JIRA 和 GitHub 平台的 Hadoop

和 VSCode 数据集进行跨平台性能评估,实验结果如表 5 和表 6.由于平台数据中元数据字段存在大量缺失(Hadoop 缺少 product 字段,且部分报告无 component 和 version 字段;VSCode 缺失 product、component、priority 字段,且大量报告缺少 version 字段),除 BERT-MLP 外,其余多特征融合方法的性能均受到显著影响.结果表明,本文方法在跨平台场景下依旧有效,在 Hadoop 和 VSCode 项目上相比于文献[17]所提方法,F1 平均提升 4.85%和 1.54%,Accuracy 平均提升 4.03%和 1.04%.

表 5　Hadoop 数据集对比结果

Table 5　Comparison results on Hadoop dataset　　单位:百分比(%)

方法	MLP				KNN				Linear SVM				Random Forest			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
BERT-MLP	79.86	90.21	79.34	83.35	79.24	90.52	79.15	83.52	77.35	90.61	77.27	81.06	77.36	90.35	77.01	81.67
D_TS	88.84	87.11	88.84	87.78	60.52	83.90	60.52	68.26	80.87	85.81	80.87	82.96	88.38	72.69	89.02	80.30
文献[17]方法	90.44	88.34	90.44	88.26	68.84	83.52	68.84	73.53	81.80	86.02	81.80	83.61	89.14	88.79	90.75	88.96
本文方法	**89.43**	**89.01**	**90.96**	**89.24**	**79.87**	**89.08**	**83.05**	**85.96**	**85.40**	**86.90**	**85.40**	**86.04**	**91.62**	**94.60**	**91.63**	**92.51**

表 6　VSCode 数据集对比结果

Table 6　Comparison results on VSCode dataset　　单位:百分比(%)

方法	MLP				KNN				Linear SVM				Random Forest			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
BERT-MLP	83.22	86.62	83.35	84.16	72.52	85.15	72.83	76.09	76.00	85.12	76.84	79.15	75.87	84.28	75.67	78.63
D_TS	81.15	85.46	81.15	82.81	68.88	83.52	68.84	73.53	85.40	86.90	85.40	86.04	85.57	90.19	85.57	87.02
文献[17]方法	87.92	87.46	87.92	87.67	68.86	83.54	68.90	73.57	87.36	88.07	87.56	87.79	89.35	92.27	89.35	90.23
本文方法	**88.56**	**91.48**	**88.56**	**89.48**	**68.98**	**83.53**	**68.95**	**73.61**	**89.84**	**92.29**	**89.84**	**90.61**	**90.24**	**93.79**	**89.76**	**91.73**

3.4　各模块特征性能分析

为验证各个模块对方法性能的影响,设置仅使用 Sliding Window Attention 的 Longformer 模型得出的语义相似度特征作为分类模型的唯一输入,依次加入融合 BiLSTM Attention 机制的语义相似度特征(+BiLSTM Attention)、元数据相似度特征(+Category Feature)和词组重叠特征(+Overlap Feature)进行模块影响分析实验,在 Eclipse、NetBeans、OpenOffice、Hadoop 和 VSCode 项目上的结果如表 7 至表 11.

表 7　Eclipse 数据集结果

Table 7　Experimental results on Eclipse dataset　　单位:百分比(%)

方法	MLP				KNN				Linear SVM				Random Forest			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
Baseline	85.29	87.61	80.15	82.28	83.33	85.55	77.65	79.70	82.92	87.96	75.98	78.35	84.26	86.18	79.03	81.04
+BiLSTM Attention	90.71	91.62	87.87	89.32	89.89	90.63	86.96	88.38	90.52	91.61	87.50	89.06	88.25	88.28	85.46	86.59
+Category Feature	93.00	93.31	91.18	92.10	92.39	92.61	90.51	91.42	92.61	92.67	90.93	91.70	92.96	93.18	91.21	92.07
+Overlap Feature	88.84	90.63	84.97	86.90	86.37	88.99	81.39	83.60	87.04	90.37	81.92	84.30	88.51	90.19	84.62	86.51
本文方法	97.05	97.11	96.37	96.72	96.30	96.60	95.24	95.86	96.80	96.94	95.99	96.44	96.93	97.22	96.02	96.58

表 8　NetBeans 数据集结果

Table 8　Experimental results on NetBeans dataset　　单位:百分比(%)

方法	MLP				KNN				Linear SVM				Random Forest			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
Baseline	70.39	78.99	71.63	68.75	70.27	77.54	71.43	68.86	65.18	78.12	66.72	61.85	70.41	77.25	71.54	69.09
+BiLSTM Attention	85.66	87.81	86.68	86.12	86.18	87.13	86.14	85.60	85.83	87.60	86.35	85.76	84.32	85.72	84.79	84.26
+Category Feature	88.10	89.41	88.55	88.06	86.56	87.97	87.03	86.52	85.16	87.30	85.74	85.06	89.45	90.25	89.81	89.44
+Overlap Feature	76.67	81.33	77.55	76.12	72.70	79.22	73.78	71.64	65.64	76.91	67.10	62.74	76.61	81.38	77.50	76.05
本文方法	93.85	94.65	94.68	94.49	94.49	94.04	94.05	93.85	93.68	93.96	93.91	93.68	94.60	94.73	94.78	94.59

表 9 OpenOffice 数据集结果

Table 9 Experimental results on OpenOffice dataset 单位:百分比(%)

方法	MLP				KNN				Linear SVM				Random Forest			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
Baseline	73.35	77.98	77.54	73.34	72.33	76.47	76.27	72.33	71.07	77.29	75.85	71.01	72.32	76.44	76.25	72.31
+BiLSTM Attention	86.96	88.28	90.24	88.40	88.58	86.99	88.92	86.80	88.34	88.10	90.06	88.16	87.13	87.12	89.05	86.97
+Category Feature	81.40	82.53	83.99	81.30	78.53	80.56	81.57	78.48	80.90	81.94	83.41	80.80	81.64	82.79	84.26	81.55
+Overlap Feature	79.35	81.44	82.47	79.31	75.75	78.97	79.34	75.74	76.79	80.23	80.53	76.79	79.14	81.22	82.24	79.11
本文方法	94.57	93.89	95.15	94.38	92.77	92.09	93.88	92.58	94.19	93.49	95.08	94.02	93.18	92.51	94.29	93.01

表 10 Hadoop 数据集结果

Table 10 Experimental results on Hadoop dataset 单位:百分比(%)

方法	MLP				KNN				Linear SVM				Random Forest			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
Baseline	77.06	79.89	76.69	78.26	74.23	77.05	75.27	76.15	74.27	78.89	76.32	77.58	74.87	77.98	72.22	74.99
+BiLSTM Attention	91.30	94.21	91.30	92.19	89.15	93.29	89.15	90.44	92.29	94.52	92.29	92.98	84.61	92.16	84.61	86.96
+Category Feature	77.30	85.06	77.30	80.48	72.54	75.91	72.54	74.19	73.26	84.84	73.26	77.75	76.61	86.96	70.52	77.88
+Overlap Feature	78.79	82.71	80.54	81.61	75.84	76.01	74.79	75.41	76.79	80.23	80.53	80.38	89.81	93.98	89.81	91.05
本文方法	89.43	89.01	90.96	89.24	79.87	89.08	83.05	85.96	85.40	86.90	85.40	86.04	91.62	94.60	91.63	92.51

表 11 VSCode 数据集结果

Table 11 Experimental results on VSCode dataset 单位:百分比(%)

方法	MLP				KNN				Linear SVM				Random Forest			
	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
Baseline	79.21	83.25	77.54	80.29	75.72	78.23	74.23	76.18	76.59	79.33	76.26	77.76	72.87	75.98	70.22	72.99
+BiLSTM Attention	86.11	89.72	86.11	87.32	79.18	88.28	79.18	81.94	85.01	89.58	85.01	86.49	85.77	90.34	85.77	87.19
+Category Feature	72.31	76.42	73.58	74.97	68.96	83.51	68.96	73.62	76.84	78.49	76.25	77.35	74.00	86.97	74.00	77.83
+Overlap Feature	80.21	84.63	80.21	81.95	74.55	77.94	76.26	77.09	87.58	90.77	87.58	88.62	89.81	93.98	89.81	91.05
本文方法	88.56	91.48	88.56	89.48	68.98	83.53	68.95	73.61	89.84	92.29	89.84	90.61	90.24	93.79	89.76	91.73

以上五组实验结果表明,三个模块抽取的相似度特征均有贡献,尤其是四种元数据类别特征,仅需配合 Baseline 抽取的语义相似度特征便可有效提高检测性能,在 Eclipse、NetBeans、OpenOffice 项目上 F1 平均提升 8.79%、18.26%、8.35%,Accuracy 平均提升 11.48%、19.88%和 8.28%,验证了多特征融合提升检测性能的可行性.但是由于 Hadoop 和 VSCode 数据普遍缺乏元数据信息,因此加入 Category Feature 模块后反倒引入了低判别能力的特征,导致在 Hadoop 和 VSCode 项目上 F1 平均提升 0.83%、降低 0.87%,Accuracy 平均降低 0.18%、3.07%.其次,语义归纳层 BiLSTM Attention 的融合也进一步提升了语义相似度特征的准确性,仅通过单个特征就能够在 Eclipse、NetBeans、OpenOffice、Hadoop 和 VSCode 项目上平均提升 5.89%、16.41%、15.48%、13.89%和 8.93%的 F1 精度,7.50 %、18.05%、15.33%、14.23%和 7.92%的 Accuracy.结合对比实验中 BERT-MLP 的结果,可以发现采用强化长程依赖信息的 Longformer 模型能够利用更多有效信息,从而达到较于 BERT 模型更优的性能表现.最后,三种词组重叠特征的引入为 Eclipse、NetBeans、OpenOffice、Hadoop 和 VSCode 项目平均提升 3.74%、3.85%、5.24%、5.36%和 7.87%的 F1 精度,4.99%、7.74%、5.24%、5.20%和 6.94%的 Accuracy.尽管该类特征影响较弱,但也能从文本比对的角度提供更多样性和更丰富的差异信息,同时也验证了传统信息检索方式与现阶段深度学习方式抽取的特征经过融合后一同用于检测重复项的有效性.

4 结束语

本文针对现阶段研究中特征组合单一问题和 BERT 类模型输入长度的局限性,提出一种强化文本长程依赖和多特征融合的重复软件缺陷报告检测方法.该方法一方面通过结合 Longformer 和 BiLSTM-Attention 实现了篇章级文本特征提取,避免表征过程中

大量信息遗失的情况.另一方面,结合语义相似度特征、元数据相似度特征以及词组重叠度特征对软件缺陷报告进行更加全面的重复性检测.在大规模公开数据集上进行实验,结果表明多特征融合的方式能够显著提升检测精度,并且强化语义差异表示后生成的语义相似度能够在复杂样本中提供更加稳定的增益,从而在跨平台场景下依旧有效.在未来的工作中,将从分类检测阶段入手,在集成分类和多级检测方向进行探索,尝试进一步提升方法的性能.

参考文献

[1] MORAN K, LINARES-VÁSQUEZ M, BERNAL-CÁRDENAS C, et al. FUSION: A tool for facilitating and augmenting Android bug reporting [C]//Proceedings of the 38th International Conference on Software Engineering Companion. Austin: ACM, 2016: 609-612.

[2] SERRANO N, CIORDIA I. Bugzilla, ITracker, and other bug trackers [J]. IEEE Software, 2005, 22(2): 11-13.

[3] RUNESON P, ALEXANDERSSON M, NYHOLM O. Detection of duplicate defect reports using natural language processing [C]//29th International Conference on Software Engineering (ICSE'07). Minneapolis: IEEE, 2007: 499-510.

[4] WANG X Y, ZHANG L, XIE T, et al. An approach to detecting duplicate bug reports using natural language and execution information [C]//Proceedings of the 13th International Conference on Software Engineering - ICSE '08. Leipzig, Germany: ACM, 2008: 461.

[5] SUN C N, LO D, WANG X Y, et al. A discriminative model approach for accurate duplicate bug report retrieval [C]//Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1. Cape Town: ACM, 2010: 45-54.

[6] AGGARWAL K, RUTGERS T, TIMBERS F, et al. Detecting duplicate bug reports with software engineering domain knowledge [C]//2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER). Montreal: IEEE, 2015: 211-220.

[7] SUN C N, LO D, KHOO S C, et al. Towards more accurate retrieval of duplicate bug reports [C]//2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011). Lawrence: IEEE, 2011: 253-262.

[8] NGUYEN A T, NGUYEN TT, NGUYEN T N, et al. Duplicate bug report detection with a combination of information retrieval and topic modeling [C]//Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering. Essen: ACM, 2012: 70-79.

[9] SUREKA A, JALOTE P. Detecting duplicate bug report using character N-gram-based features [C]//2010 Asia Pacific Software Engineering Conference. Sydney: IEEE, 2010: 366-374.

[10] SABOR KK, HAMOU-LHADJ A, LARSSON A. DURFEX: A feature extraction technique for efficient detection of duplicate bug reports [C]//2017 IEEE International Conference on Software Quality, Reliability and Security (QRS). Prague: IEEE, 2017: 240-250.

[11] WANG JJ, LI M Y, WANG S, et al. Images don't lie: Duplicate crowdtesting reports detection with screenshot information [J]. Information and Software Technology, 2019, 110: 139-155.

[12] DESHMUKH J, ANNERVAZ K M, PODDER S, et al. Towards accurate duplicate bug retrieval using deep learning techniques [C]//2017 IEEE International Conference on Software Maintenance and Evolution (ICSME). Shanghai: IEEE, 2017: 115-124.

[13] AKILAN T, SHAH D, PATEL N, et al. Fast detection of duplicate bug reports using LDA-based topic modeling and classification [C]//2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC). Toronto: IEEE, 2020: 1622-1629.

[14] XIE Q, WEN Z Y, ZHU J M, et al. Detecting duplicate bug reports with convolutional neural networks [C]//2018 25th Asia-Pacific Software Engineering Conference (APSEC). Nara: IEEE, 2018: 416-425.

[15] 周文杰,谢琪,崔梦天.强化文本关联语义和多特征提取的重复缺陷报告检测模型[J].重庆大学学报,2023,46(7):53-62.

[16] LAZAR A, RITCHEY S, SHARIF B. Improving the accuracy of duplicate bug report detection using textual similarity measures [C]//Proceedings of the 11th Working Conference on Mining Software Repositories. Hyderabad: ACM, 2014: 308-311.

[17] 曾杰,贲可荣,张献,等.融合文本分布式表示的重复缺陷报告检测[J].计算机工程与科学,2021,43(4):670.

[18] 曾方,谢琪,崔梦天.一种融合 D_BBAS 方法的重复缺陷报告检测[J].计算机应用研究,2022,39(12):3736-3742. DOI: 10.19734/j.issn.1001-3695.2022.05.0241.

[19] BEN MESSAOUD M, MILADI A, JENHANI I, et al. Duplicate bug report detection using an attention-based neural language model [J]. IEEE Transactions on Reliability, 2023, 72(2): 846-858.

[20] ISOTANI H, WASHIZAKI H, FUKAZAWA Y, et al. Duplicate bug report detection by using sentence embedding and fine-tuning [C]//2021 IEEE International Conference on Software Maintenance and Evolution (ICSME). Luxembourg: IEEE, 2021: 535-544.

[21] JIN Y, ZHU Q S, DENG X, et al. Weighted hierarchy mechanism over BERT for long text classification [C]// Artificial Intelligence and Security. Cham: Springer International Publishing, 2021: 566-574.

[22] BELTAGY I, PETERS M E, COHAN A. Longformer: The long-document transformer [EB/OL]. 2020: 2004.05150. https://arxiv.org/abs/2004.05150v2.

[23] Šarić F, Glavaš G, Karan M, et al. Takelab: Systems for measuring semantic text similarity [C]// * SEM 2012: The First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012). 2012: 441-448.

[24] ZHANG T, HAN D, VINAYAKARAO V, et al. Duplicate bug report detection: How far are we? [J]. ACM Transactions on Software Engineering and Methodology, 2023, 32(4): 1-32.

(责任编辑:张阳,殷锋,付强,和力新,肖丽;英文编辑:周序林,郑玉才)