计算机科学
COMPUTER SCIENCE

融合双向门控循环单元和注意力机制的软件自承认技术债识别方法

熊罗庚，郑尚，邹海涛，于化龙，高尚

引用本文

熊罗庚，郑尚，邹海涛，于化龙，高尚. 融合双向门控循环单元和注意力机制的软件自承认技术债识别方法[J]. 计算机科学, 2022, 49(7): 212-219.

XIONG Luo-geng, ZHENG Shang, ZOU Hai-tao, YU Hua-long, GAO Shang. Software Self-admitted Technical Debt Identification with Bidirectional Gate Recurrent Unit and Attention Mechanism [J]. Computer Science, 2022, 49(7): 212-219.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.210500075

融合双向门控循环单元和注意力机制的软件自承认技术债识别方法

熊罗庚　郑　尚　邹海涛　于化龙　高　尚
江苏科技大学计算机学院　江苏 镇江 212100
(xlg935003328@sina.com)

摘　要　软件自承认技术债(Self-admitted Technical Debt,SATD)由程序开发人员写入项目的源代码注释中,是开发人员为追求短期效益而刻意留下软件缺陷的说明,大量的SATD将不利于软件维护。近年来,越来越多的学者致力于软件SATD识别的研究,并提出了不同的识别方法,如基于自然语言处理或文本挖掘等检测方法。然而,大多数研究结果依赖于现有的词库或手工提取的特征,不仅耗费了大量的时间,而且增加了计算复杂度,识别结果并不理想。基于此,提出了一种基于双向门控循环单元(Gate Recurrent Unit,GRU)和注意力机制的软件自承认技术债识别方法,通过Word2vec中的Skip-gram模型获取词向量,构建双向GRU网络获取高级特征,并利用注意力机制自动发现对SATD分类起到关键作用的词,从而捕获最重要的语义信息。实验结果表明,本文方法在精确率、召回率和F1-score上均有较优表现,能够有效地识别软件SATD,避免了传统任务中复杂的特征工程。

关键词:软件维护;自承认技术债;Word2vec;注意力机制;GRU
中图法分类号　TP311

Software Self-admitted Technical Debt Identification with Bidirectional Gate Recurrent Unit and Attention Mechanism

XIONG Luo-geng,ZHENG Shang,ZOU Hai-tao,YU Hua-long and GAO Shang
School of Computer,Jiangsu University of Science and Technology,Zhenjiang,Jiangsu 212100,China

Abstract　Software self-admitted technical debt(SATD) is written into the source code comments of the project by developers who leave a note admitting incurring intentionally for short-term benefits,and a large amount of SATD will be dangerous to software maintenance. In recent years,more scholars focus on the research of software SATD recognition and propose different identification approaches,such as SATD detection based on natural language processing or text mining. However,the identification results of most previous studies are not very well due to the existing thesaurus or manually extracted features,which not only consumes a lot of time,but also increases computational complexity. Therefore,a software SATD identification approach based on bidirectional gated recurrent unit(GRU) and attention mechanism is proposed. The word vector is obtained first through the Skip-gram model,and the bidirectional GRU network is constructed to obtain the high-level features. Finally,the attention mechanism is used to automatically discover words that play a key role in SATD identification,and the most important semantic information can be captured. Experimental results show that the proposed approach has excellent performance in precision,recall and F1-score. It can effectively identify software SATD and avoid complex feature engineering in traditional tasks.

Keywords　Software maintenance,SATD,Word2vec,Attention mechanism,GRU

1　引言

在软件开发过程中,开发人员始终在软件代码质量和软件发布时间这两者间进行权衡[1],一种用于描述该现象的隐喻被称作技术债[2]。然而,过多的技术债会对软件长期的健康发展造成不可预知的影响[3]。其中,自承认技术债(SATD)是技术债的一种特殊情况[4]。自承认技术债由开发人员在源代码的注释中注明,并且被认为是发现软件

到稿日期:2021-05-12　返修日期:2021-09-06
基金项目:江苏省高等学校自然科学研究面上基金(18JBK520011);江苏省镇江市重点研发计划(社会发展)项目(SH2019021);江苏省自然科学基金面上项目(BK20191457)
This work was supported by the Natural Science Research Foundation for Higher Education of Jiangsu Province(18JBK520011),Primary Research and Development Plan(Social Development) of Zhenjiang(SH2019021) and Natural Science Foundation of Jiangsu Province(BK20191457).
通信作者:郑尚(szheng@just.edu.cn)

缺陷特征的重要来源[5]。

近年来，随着软件产品规模不断扩大，对软件自承认技术债的识别成为了热门的研究课题。Sierra 等[6]分析了当前的 SATD 检测和偿还方法以及其技术特点；Zampetti 等[7]分析了多个软件在不同版本下的 SATD 的变化情况；Aversano 等[8]将 SATD 划分为不同的种类。SATD 识别本质上是一个二分类问题，多数学者的关注点在于特征的提取和分类模型的构建。例如，Huang 等[9]通过不同项目的代码注释来构建多个贝叶斯分类器，利用集成思想进行投票预测；Jernej[5]等通过词向量的相似度来扩充文本特征，提高了分类效果。Huang 等[3]采用交叉过采样的方法解决了技术债类别不平衡的问题；Maldonado 等[10]使用自然语言处理的方法识别 SATD；Potdar 等[4,11]使用人工检查的方式检测潜在的 SATD，他们确定了 62 种模式用于识别 SATD；Wehaibi 等[12]使用上述 62 种模式对 SATD 识别展开了进一步的分析。

然而，代码注释中的 SATD 固有的特性给基于模式或传统文本挖掘的 SATD 识别的准确性带来了巨大的影响。SATD 固有特性及带来的影响如下。

(1)词汇多样性。开发人员经常使用“todo”“fixme”“hack”等术语来表示 SATD，然而对于很多不明显的术语，如“should”“perhaps”“may be”等。基于模式或文本挖掘方法可能会过滤并删除类似词汇，同时词汇的多样性会导致词汇量过大的现象，传统的方法通常采用词袋模型进行词向量表示，造成了严重的向量稀疏问题。

(2)语义变异。一些语义相似的 SATD 评论可能会以不同的方式表达，如“should not really well”和“not totally well”。同时，相同的词可能表示 SATD 或不表示 SATD，具体取决于整体评论上下文。传统的文本挖掘方法或基于模式的方法可能会遗漏语义相似且表达方式不同的评论，或将非 SATD 评论错误地归类为 SATD 评论。

本文的主要贡献如下：

(1)本文利用 Word2vec 中的 Skip-gram 模型进行词向量表示，生成低维稠密的向量，避免了词汇量过大带来的向量稀疏问题。

(2)融合双向 GRU 网络和注意力机制构建 SATD 识别模型。其中，GRU 层使用双向网络从嵌入层获取高级特征，注意力层则通过权重相乘获取句子级的特征，使得提取的特征带有关键的 SATD 信息。此模型不依赖基于模式或传统文本挖掘的方法，通过上下文学习对 SATD 识别具有重要作用的信息，避免词汇遗漏，提高识别准确度。

在开源数据集上进行了验证，结果表明，本文方法的 Precision，Recall 和 F1-Score 均达到 80%以上，且高于现有研究方法。

2 基础知识

2.1 Word2vec

已有研究采用的词袋模型[3-4,9]导致词向量变得高维稀疏。与词袋模型(Bag of Word，BOW)相比，Word2vec 生成的词向量模型[13]通过词嵌入将 word 从高维稀疏向量映射到低维稠密向量，能够解决维数灾难和语义相关问题。假设给定一个由 T 个单词组成的语句 $S=\{w_1,w_2,\cdots,w_T\}$，对于 S 中的每个单词 w_i，先将其转化为独热向量 $\boldsymbol{e}_i$，随后使用嵌入矩阵 $\boldsymbol{W}^{\text{word}}$ 将其转化为最终词向量 $\boldsymbol{v}_i$，即 $\boldsymbol{v}_i=\boldsymbol{e}_i\cdot\boldsymbol{W}^{\text{word}}$。

Word2vec 包含 CBOW 模型与 Skip-gram 模型。由 Mikolov 等[13]的结论可知，Skip-gram 模型的训练时间比 CBOW 模型稍长，但性能表现更好，且 CBOW 模型在遇到生僻词时性能下滑明显。鉴于不同开发人员的代码注释方式不同，本文将选用 Skip-gram 模型进行词向量表示。

2.2 门控循环单元与双向网络结构

由于循环神经网络(Recurrent Neural Network，RNN)结构简单，导致在处理长序列输入时会出现梯度消失等问题。因此，研究人员对 RNN 进行了相应的改进，其中最具代表性的工作是长短期记忆网络(Long Short-Term Memory，LSTM)[14]和门控循环单元[15-17]。与 LSTM 相比，GRU 内部参数少，易于计算，且过拟合风险小，训练速度更快。

然而，由于单向网络不能较好地处理上下文词汇关系，Schuster 等[18]提出了双向网络结构(Bidirectional Network)，构建了两个独立且相同的神经网络，分别学习目标词汇与上下文的依赖关系，并将所得结果进行拼接，得到最终的输出结果。根据表 1 中软件 SATD 的文本描述，代码注释中与 SATD 有关的词汇为“should”，如果不考虑上下文的依赖关系，将会造成相关特征的丢失等问题，因此双向网络结构更有利于软件 SATD 的识别。

表 1 代码注释中的 STAD 词汇

Table 1 STAD words in code comments

序号	SATD 示例
1	hack for to do items only, ***should*** check isLeaf (node), but that, includes empty folders
2	Instead that subsystem ***should*** register its desired menus and actions
3	TODO: this class ***should*** be moved to package
4	the new element ***should*** be created
5	We probably ***should*** do the whole traversal in a single MDR transaction
…	…

2.3 注意力机制

注意力机制提高了神经网络的可解释性，并且在一定程度上帮助 RNN 解决了梯度消失的问题。近年来，结合注意力机制的神经网络在多个领域(如情感分析、语义抽取、语音图像识别等)均取得了显著的成果。例如，Peng 等[19]将注意力机制与 LSTM 网络相结合进行文本关系分类，并取得了良好的效果；Wang 等[20]使用注意力机制改善了语义关系抽取的效果。

根据双向 GRU 层获取的高级特征，本文通过注意力机制生成权重向量，将词汇级特征合并为句子级的特征，使得整个模型能够捕获最重要的 SATD 语义信息以提高 SATD 的识别准确率。

3 问题定义

SATD 于 2014 年被首次提出[4]，是项目开发人员所承认

的软件问题，如代码问题、缺陷及其产生的原因和解决方案等。开发人员有意地将这些信息添加到项目中，用于临时满足开发需求，并加以标注以便后期将其移除。

根据 SATD 的定义，学术界将 SATD 的识别研究转化为二分类任务的形式化流程[6-7,21-23]，其主要流程为数据收集、文本预处理、模型构建和模型验证这 4 个基本步骤。但是，SATD 的固有特性使得其识别与传统的文本二分类任务不同，其对模型的泛化能力要求更高。

因此，本文要解决以下两个方面的问题：

(1)如何避免因词汇量大而造成的高维稀疏，使其节省计算开销；

(2)以词向量为输入，如何通过上下文学习来获取代码注释中的 SATD 重要信息，避免词汇遗漏，提高识别准确率。

4 SATD 识别方法描述

本文方法的总体框架如图 1 所示，包含模型训练和模型验证两个阶段。其中，模型训练包含源代码注释收集、文本预处理和模型构建这 3 个步骤。模型构建为本文方法的关键部分；模型验证包含目标代码注释收集、文本预处理和标签预测这 3 个步骤。下文将分别对训练阶段和验证阶段展开描述。

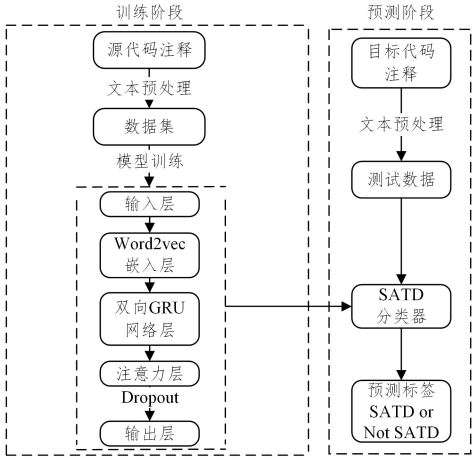

图 1 识别方法的总体框架

Fig. 1 Overall framework of identification method

4.1 模型训练

SATD 识别模型的训练步骤如算法 1 所示。

算法 1 模型训练阶段

输入：源代码注释

输出：SATD 分类器

步骤 1 源代码的 SATD 注释收集；

步骤 2 文本预处理；

步骤 3 SATD 分类器构建。

算法 1 中，源代码注释能够通过 Doxgen[24] 来提取。本文采用公开的 SATD 注释集，因此步骤 1 未有过多描述。

4.1.1 文本预处理

由于获取的软件 SATD 注释包含大量的无用信息，为了提取有用词组，本文采用以下步骤对源代码注释进行预处理。

(1)令牌化：将文本中的非单词类的符号删除，仅保留英文单词；随后根据空格切分文本，将文本转化为多个英语单词；最后将所有英语单词小写。

(2)去除停用词：本文去除了常用的停用词，如“the”“for”“be”等。

(3)词根还原：该过程将单词的变体转化为其词干、基数形式或词根。本文采用词根还原工具 Porter stemmer2 以完成这项工作。

4.1.2 SATD 分类器的构建

对文本进行预处理后，需构建 SATD 分类器，其为识别模型训练阶段的关键部分。如图 2 所示，本文构建的分类器网络结构包含以下 5 层。

(1)输入层：将预处理后的文本输入到 Embedding 层。

(2)Embedding 层：接收输入层的文本，为每个单词生成对应的词向量。

(3)双向 GRU 层：接收嵌入层的词向量，学习词汇依赖关系，通过正向 GRU 和反向 GRU 进行特征联合，生成高级词特征。

(4)注意力层：生成一个权重向量，通过与权重向量相乘的方式将词特征组合为句子特征。

(5)输出层：将上述得到的特征通过 sigmoid 函数计算，从而得到每个目标代码注释的 SATD 预测概率。

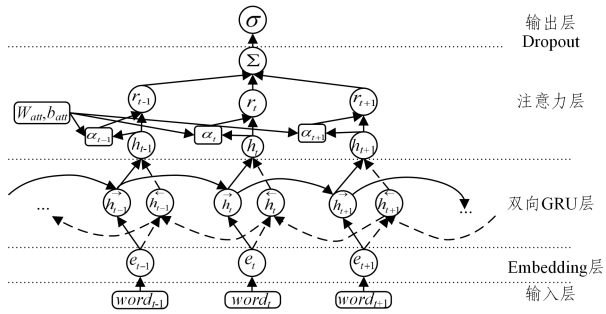

图 2 SATD 分类器模型结构

Fig. 2 Model structure of SATD classifier

其中，核心层的具体描述如下。

(1)Embedding 层

根据输入层的文本，使用 Skip-gram 模型为每个单词生成一个词向量 v，则每条代码注释被表示为 $\{v_1, v_2, \cdots, v_N\}$。由于 GRU 网络要求输入序列的长度一致，因此本文截取每个文本的前 N 个单词作为输入。若文本总长度小于 N，则使用空单词填充至 N。同时，为去除空单词对文本语义的干扰，本文使用掩码(Mask Code)来阻止空单词参与模型训练。

(2)双向 GRU 层

GRU 的内部结构如图 3 所示，其具体工作流程如下。

1)构建重置门：由历史信息和当前输入生成重置门 $r^{(t)}$：

$$r^{(t)}=\sigma(W^{(r)}\cdot x^{(t)}+U^{(r)}\cdot h^{(t-1)}) \tag{1}$$

2)构建更新门：由历史信息和当前输入生成更新门 $z^{(t)}$：

$$z^{(t)}=\sigma(W^{(z)}\cdot x^{(t)}+U^{(z)}\cdot h^{(t-1)}) \tag{2}$$

3)信息重置：由重置门对上文信息进行选择，随后将结果整合至当前输入中，得到隐藏状态 $\tilde{h}^t$。重置门主要决定需要被遗忘的上文信息。

$$\tilde{h}^{(t)}=\tanh(W\cdot x^{(t)}+r^{(t)}\circ(U\cdot h^{(t-1)})) \tag{3}$$

其中，$\circ$为 Hadamard Product。

4）信息更新：由更新门对上文信息和隐藏状态进行选择，并整合得到单元输出。更新门主要决定需要被继续传递的信息。

$$h^{(t)}=\tanh(z^{(t)}\circ h^{(t-1)}+(1-z^{(t)})\circ\tilde{h}^{(t)}) \tag{4}$$

$$y^{(t)}=h^{(t)} \tag{5}$$

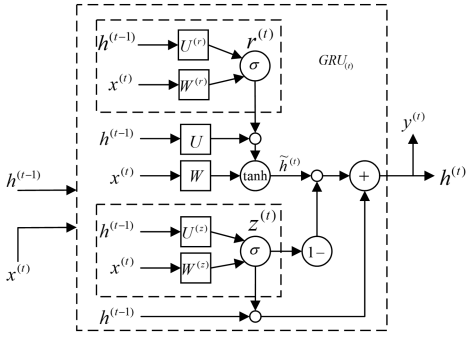

图 3　GRU 内部结构

Fig. 3　GRU internal structure

由于单向 GRU 仅能捕获上文信息，而对于每个时序而言，上文信息与下文信息同样重要。因此，SATD 采用双向 GRU，通过正反两个方向同时捕获每个时序的上下文信息。双向 GRU 层的最终输出由两层 GRU 网络输出拼接得到，如式(6)所示：

$$h_t=[\overrightarrow{h}_t\oplus\overleftarrow{h}_t] \tag{6}$$

其中，$\oplus$为连接符。

根据式(6)，将 Embedding 层所获得的$\{v_1,v_2,\cdots,v_n\}$输入到双向 GRU 网络层中，通过正向 GRU 获取正向特征向量$\{\overrightarrow{h}_1,\overrightarrow{h}_2,\cdots,\overrightarrow{h}_t\}$，通过反向 GRU 获取反向特征向量$\{\overleftarrow{h}_1,\overleftarrow{h}_2,\cdots,\overleftarrow{h}_t\}$，并将二者进行连接组合，则每条代码注释被表示为$\{h_1,h_2,\cdots,h_t\}$，$t=1,2,3,\cdots,n$。

(3)注意力层

注意力层接收双向 GRU 输出的组合特征，使用权重向量$\boldsymbol{W}_{\text{att}}$与偏置项$b_{\text{att}}$为每个特征生成一个权重值$\alpha_t$，得到不同特征的重要性。最后，将权重与特征加权求和得到句子特征o。根据双向 GRU 网络层的代码注释输出$\{h_1,h_2,\cdots,h_t\}$，注意力层按照式(7)－式(9)得到句子特征o。

$$M=\tanh(\boldsymbol{M}_{\text{att}}\cdot H+b_{\text{att}}) \tag{7}$$

$$\alpha_t=\frac{\exp(M_t)}{\sum_t\exp(M_t)},t=1,2,\cdots,n \tag{8}$$

$$o=\sum_t\alpha_t\cdot h_t,t=1,2,\cdots,n \tag{9}$$

(4)输出层

输出层由单个神经元组成，与注意力层全连接，激活函数为 sigmoid，该单元的输出值将作为判定某注释是否为 SATD 的标准，若输出值大于等于 0.5，则认为是 SATD；否则不是 SATD。激活函数 sigmoid 的计算式如下：

$$\sigma(x)=\frac{1}{1+\exp(-x)} \tag{10}$$

为减小模型的过拟合程度，SATD 在注意力层与输出层之间应用了 Dropout 算法。该算法的核心思想是，在神经网络进行前向传播时，使每个神经元有一定概率 P 停止工作，以此提高模型的泛化能力，从而减小对局部特征的依赖。

网络的损失函数为二分类交叉熵损失函数，其公式可以简化为：

$$Loss=-\sum_i^n(y_i\times\log(\hat{y}_i)+(1-y_i)\times\log(1-\hat{y}_i))+\lambda\parallel\theta\parallel_F^2 \tag{11}$$

其中，y_i 为样本的真实标签，$\hat{y}_i$ 为分类器输出，λ 为 L2 正则化系数。

最后，训练上述基于双向 GRU 和注意力机制模型的相关参数，当达到最大迭代参数时，即可得到分类器模型。为了提高模型的性能，本文使用 Adam 优化算法[17]，与传统的随机梯度下降优化算法相比，能够为不同参数提供自适应学习率，以保证网络的稳定性。

4.2　模型验证

根据 4.1 节生成的 SATD 分类器，本节通过算法 2 描述的步骤来完成模型验证，以确保模型的准确性。

算法 2　模型验证阶段

输入：目标代码注释

输出：SATD 预测概率

步骤 1　新的代码注释进行文本预处理；

步骤 2　预处理后输入到 SATD 分类器；

步骤 3　获取预测标签。

5　实验结果与分析

5.1　数据集描述

本文涉及的所有实验均在运行 Microsoft Windows 10 的四核 CPU(E3-1231 v3，3.4 GHz)、16 GB 内存和 512 GB 硬盘的 PC 上实现，并采用了已有研究者[4]所公布的数据集。在实验中，对数据集进行随机划分，其中训练集占 70%，测试集占 30%。同时，由于神经网络初始化具有随机性，因此实验结果取 100 次实验的平均值。

本文实验使用的数据集如表 2 所列，其中 SATD 的占比较低，属于较为典型的类别不平衡数据集。本文通过在模型训练过程中为两个样本类别提供不同的训练权重系数 k 来弱化这一影响，权重系数将在后续描述中讨论。

表 2　数据集

Table 2　Dataset

Project	Comments	Numbers of SATD	SATD/%
ArgoUML	5426	969	17.9
Columba	4090	128	3.1
Hibernate	2492	377	15.1
JEdit	4644	195	4.2
JFreeChart	2494	101	4.0
JMeter	4148	282	6.8
JRuby	3652	383	10.5
SQuirrel	4473	201	5.0

5.2　度量指标

SATD 识别实际是二分类任务，本文将 SATD 注释视为正例，非 SATD 视为反例。因此，分类结果有以下 4 种。

(1)真正例(True Positive，TP)：实际是 SATD 且被正确分类为 SATD 的样本。

(2)假正例(False Positive，FP)：实际是非 SATD 且被

错误分类为 SATD 的样本。

(3)真反例(True Negative,TN):实际是非 SATD 且被正确分类为非 SATD 的样本。

(4)假反例(False Negative,FN):实际是 SATD 且被错误分类为非 SATD 的样本。

综上所述,SATD 分类任务的混淆矩阵如表 3 所列。

表 3 SATD 混淆矩阵

Table 3 SATD confusion matrix

Actual	Predict	
	True	False
True	*TP*	*FN*
False	*FP*	*TN*

本文采用精确率、召回率和 F1-score 作为模型的性能测试指标,指标的具体描述如下。

精确率(precision,P):被正确分类的真实正例样本数量与所有被分类为正的样本数量的比例。

$$precision=\frac{TP}{TP+FP} \tag{12}$$

召回率(recall,R):被正确分类的真实正例样本数量与所有真实正例样本数量的比例。

$$recall=\frac{TP}{TP+FN} \tag{13}$$

F1 分数($F1\text{-}score$):精确率与召回率的调和平均数,可以兼顾精确率与召回率的度量。F1-score 越高,模型的性能就越好。

$$F1\text{-}score=\frac{2\times P\times R}{P+R} \tag{14}$$

当精确率较高时,模型更倾向于严格执行判定,即对于疑似 SATD 的样本有更大概率将其舍去,但可能导致某些正例被当作反例处理,使得召回率较低;当召回率较高时,模型更倾向于降低判定标准,即对于疑似 SATD 的样本有更大概率将其保留,但可能导致某些反例被当作正例处理,导致精确率较低。通常在同一模型中很难兼顾精确率和召回率,因此 $F1\text{-}score$ 能更好地反应模型的综合性能。

5.3 实验参数设置

本文方法涉及以下重要参数的设定:词库剪枝阈值 $Min\ count=5$,词向量长度 $M=100$,输入序列长度 $N=50$,正反例权重比 $k=2.25$,Dropout 概率 $P=0.5$,Adam 算法学习率 $\eta=0.0001$,批量大小 $batch\ size=32$,L2 正则化系数 $\lambda=0.005$。其中,词库剪枝阈值 $Min\ count$、词向量长度 M、Dropout 概率 P、Adam 算法学习率 η 以及批量大小 $batch\ size$ 均为默认值;对于 L2 正则化系数 λ,通过实验确定当 $\lambda=0.005$ 时,模型性能达到最佳;实验过程中发现输入序列长度 N 与权重系数比 k 的取值对模型的识别性能和计算开销有显著影响,本文将在第 6 节从 F1-score 结果和训练时间这两个方面进行综合讨论。

5.4 实验结果

本节将通过实验结果来回答第 3 节提出的两个问题。

问题 1 如何避免词汇量大造成的高维稀疏,使其节省计算开销。

由于 Skip-gram 模型能够将高维稀疏向量映射到低维稠密向量,因此本文选取其对词汇进行向量化表示。在保证相同数据集的前提下,本文比较并分析了 Skip-gram 模型和传统方法中的词袋模型各自生成词向量的时间。

如图 4 所示,当数据集变大时,BOW 生成词向量的时间快速增加;而 Skip-gram 模型生成词向量的时间增幅较为平缓。图 4 中的实验数据表明了 BOW 生成词向量的时间与数据量有指数关系,Skip-gram 生成词向量的时间与数据量成对数关系。由此可知,Skip-gram 模型有效地降低了 SATD 词向量表示的计算开销。

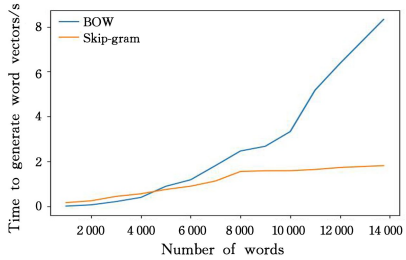

图 4 生成词向量的时间与单词数量的关系

Fig. 4 Relationship between the generation time of word vector and the numbers of words

问题 2 以词向量为输入,如何通过上下文学习获取代码注释中的 SATD 重要信息,避免词汇遗漏,提高识别准确率。

根据 4.1.2 节所述的内容,本文融合了双向 GRU 和注意力机制来构建 SATD 识别模型。将本文方法与已有研究在相同数据集上进行比较,选取的对比方法如下。

(1)贝叶斯集成(Bayes+Bagging)[6]:将信息增益作为特征提取手段,建立贝叶斯多项式分类器,利用集成思想对目标文本进行投票决策。

(2)模式识别(Pattern)[4]:将 Potdar A 和 Shihab E 总结的 62 种文本模式作为判定 SATD 的标准,当文本中出现至少 1 种文本模式时,该文本被判定为 SATD。

(3)朴素贝叶斯(NBM):根据贝叶斯公式,使用概率统计对样本数据集进行分类。

(4)支持向量机(SVM)[20]:利用支持向量(Support Vectors)进行 SATD 识别。

(5)NLP 方法[7]:基于自然语言处理,构建最大熵分类器进行 SATD 的识别。

(6)K 近邻算法(KNN)[20]:通过度量不同特征在空间中的距离,对样本进行分类。

(7)循环神经网络(RNN):使用自反馈神经元,可以处理任意长度的序列模型。

同时,为了进一步验证模型网络结构的性能,将本文方法与单双向 RNN、单双向 GRU 及注意力机制的不同组合模型进行了性能对比。

如图 5 所示,本文方法的精度达到了 85.5%,召回率达到了 83.1%,$F1$-score 达到了 84.3%。图 5 所示的实验数据表明,在 SATD 识别工作中,本文的模型在精确率、召回率与 $F1\text{-}score$ 上的表现均优于其他方法,说明本文方法具有一定的先进性和优越性。

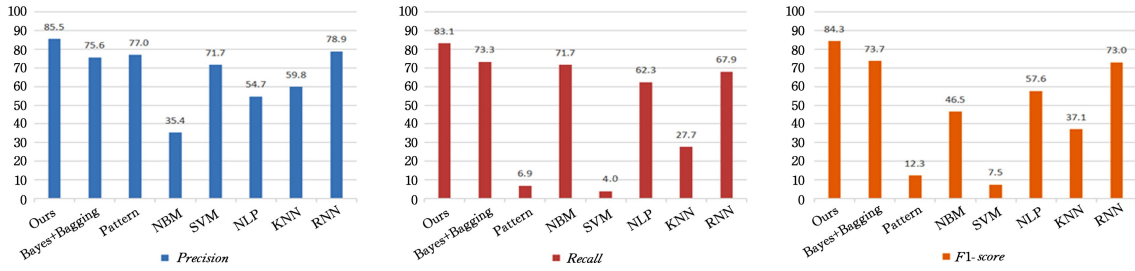

图 5 各方法的性能表现

Fig. 5 Performance of each method

图 6 给出了不同网络组合的 *Precision*，*Recall* 和 *F1-score* 的对比结果，表明了本文融合的双向 GRU 网络和注意力机制优于其他网络结构的组合，验证了以下两点：1）与 RNN 相比，GRU 避免了梯度下降的现象，能够提升 SATD 的识别性能；2）双向网络结构和注意力机制的组合能够提升单向 GRU 或 RNN 在 SATD 识别过程中的性能，说明了双向网络获取上下文信息和注意力机制来表示关键 SATD 特征的重要性。

结合图 4－图 6 可知，本文方法不仅可以避免高维稀疏的向量计算，降低计算复杂度，而且能够捕捉代码注释的上下文信息和 SATD 关键特征，提高了 SATD 的识别准确率。

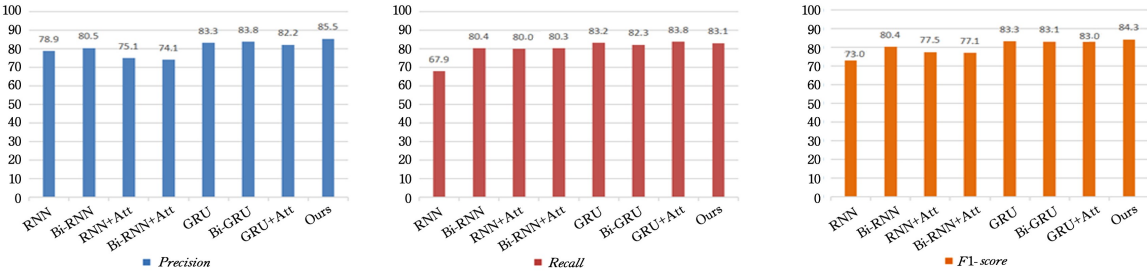

图 6 各网络的性能表现

Fig. 6 Performance of each network

6 讨论

6.1 输入序列 *N* 长度的选取

首先，本文对所用的数据集的注释长度进行了分析和统计。如图 7 所示，单词长度小于 25 的注释量占总注释量的 96.89%；单词长度为 25～50 的注释量占 2.45%；单词长度为 50～100 的注释量占 0.57%；单词长度为 100～150 的注释量占 0.07%；单词长度为 150～200 的注释量占 0.01%。

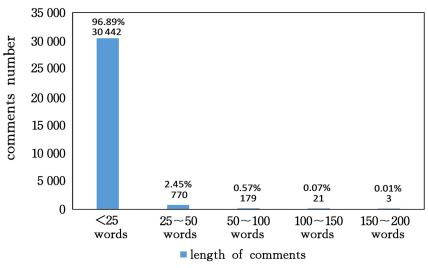

图 7 数据集注释长度分布

Fig. 7 Dataset comments length distribution

其次，本文对 SATD 关键特征在注释中出现的位置分布进行了统计和分析。如图 8 所示，本文发现 SATD 的关键特征出现在注释中前 25 个单词的数量占所有关键特征的 95.70%；出现在注释中前 50 个单词的数量占所有关键特征的 98.80%；出现在注释中前 100 个单词的数量占所有关键特征的 99.80%；出现在注释中前 150 和前 200 个单词的数量均占所有关键特征的 100%，这表明了关键特征出现在靠前单词的位置。因此，本文将在 Embedding 层提出的前 N 个单词作为输入是合理的，能够覆盖所有的关键信息，且在实验中，N 的取值范围为[0，200]。

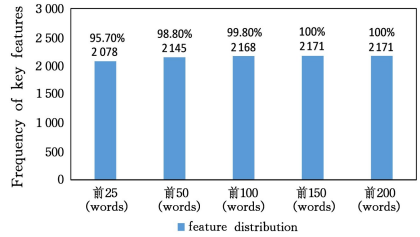

图 8 关键特征的位置分布情况

Fig. 8 Position distribution of key features

6.2 *N* 和 *k* 的取值与模型训练时间的关系

在模型构建过程中，输入序列 N 的长度和权重系数 k 将影响训练时间。如图 9 所示，当 k 的取值固定、输入序列 N 增加时，模型的训练时间呈指数级的增长趋势，其中，$N=50$ 时的训练时间为 148 s；$N=100$ 时的训练时间为 412 s；$N=150$ 和 $N=200$ 时的训练时间分别为 886 s 和 2 987 s。结合图 8 可知，虽然前 100 个单词至前 200 个单词包含更多的关键特征，但其增加了模型的训练时间，将占用较多的计算资源。

图 10 给出了当固定 N 的取值时，训练时间随权重系数 k 变化的变化趋势。结果表明，相比 k，权重系数对模型训练时间的影响较小。

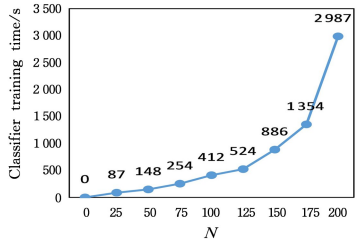

图 9 T 随 N 的变动趋势

Fig. 9 Variance of T with N

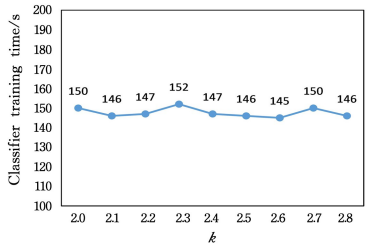

图 10 T 随 k 的变化趋势

Fig. 10 Variance of T with k

6.3 N 和 k 的选择与模型准确度的关系

在保证时间效率的前提下，本文希望模型的识别效果也能达到最优。因此，本节首先分析了 N 和 k 对 $F1$-*score* 的变化趋势，然后通过网格搜索的方法寻找最优组合，从而达到最好的识别效果。

图 11 和图 12 给出了两者的取值和 $F1$-*score* 的变化趋势。当固定 k 值时，图 11 给出了 $F1$-*score* 随 N 变化的趋势。由结果可知，并非输入序列的长度越长，$F1$-*score* 就越高，过长的输入序列反而会引进冗余的特征，降低识别准确度。

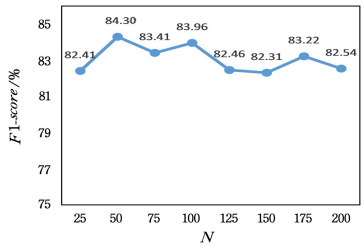

图 11 F1-score 随 N 的变化趋势

Fig. 11 Variance of F1-score with N

图 12 给出了当固定 N 值时，权重系数 k 的选取与 $F1$-*score* 的关系。由结果可知，k 值的变化也将影响识别准确度的高低。

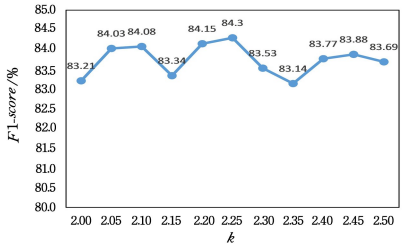

图 12 F1-score 随 k 的变化趋势

Fig. 12 Variance of F1-score with k

最后，通过网格搜索方法，图 13 给出了两者组合与 $F1$-*score* 的关系，可以看出，N 和 k 的组合变化将影响 $F1$-*score* 的表现，当 $N>50$ 时，k 发生了变化，$F1$-*score* 值有下降趋势。

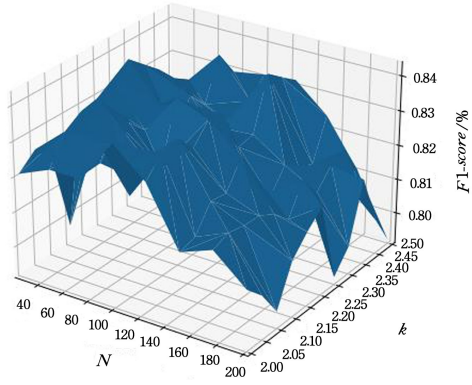

图 13 N-k 与 $F1$-*score* 的关系

Fig. 13 Relationship between N-k and $F1$-*score*

综上所述，根据对注释长度、关键特征分布、时间开销和识别准确度的分析结果，本文选取 $N=50$，$k=2.25$。此时，$F1$-*score* 为 84.3%，模型的训练时间为 148 s 秒。

结束语 本文围绕软件源代码注释中存在的自承认技术债务进行了研究。针对 SATD 识别任务，提出了融合双向 GRU 与注意力机制的 SATD 识别方法。首先，选取数据集进行文本预处理，并使用 Skip-gram 进行词嵌入，避免了计算复杂度高等问题；然后，本文构建了双向 GRU 神经网络以生成高级特征，并通过注意力层获取 SATD 的句子特征，充分考虑了上下文的语义信息；最后，在开源数据集上进行方法验证。实验结果表明，本文方法在精确率、召回率与 F1-score 上均有较优表现，能够很好地完成软件 SATD 识别。

在后续工作中，我们将在更多数据集上验证所提方法的性能，并进一步扩展模型，使其能够进行软件 SATD 的多类识别。

参 考 文 献

[1] GABRIELE B, BARBARA R. A large-scale empirical study on self-admitted technical debt[C]// Proceedings of the 13th International Workshop. IEEE, 2016: 315-326.

[2] CUNNINGHAM W. The WyCash portfolio management system [J]. Acm Sigplan Oops Messenger, 1992, 4(2): 29-30.

[3] HUANG C, XU K H, ZHENG S, et al. Software self-admitted technical debt identification approach based on cross oversampling[J]. Journal of Jiangsu University of Science and TechnologyNatural Science Edition, 2020, 182(5): 55-60.

[4] POTDAR A, SHIHAB E. An Exploratory Study on Self-Admitted Technical Debt[C]// 2014 IEEE International Conference on Software Maintenance and Evolution. IEEE, 2014: 91-100.

[5] JERNEJ F, VILI P. Enhanced Feature Selection Using Word Embeddings for Self-Admitted Technical Debt Identification [C]// Proceedings of the 2018 44th Euromicro Conference on Software Engineering and Advanced Applications(SEAA). IEEE Computer Society, 2018: 230-233.

[6] SIERRA G, SHIHAB E, KAMEI Y. A survey of self-admitted

technical debt[J]. Journal of Systems and Software,2019,152:70-82.

[7] ZAMPETTI F,SEREBRENIK A,PENTA M. Was Self-Admitted Technical Debt Removal a Real Removal? An In-Depth Perspective[C]// IEEE/ACM International Conference on Mining Software Repositories. IEEE Computer Society,2018:526-536.

[8] AVERSANO L,IAMMARINO M,CARAPELLA M,et al. On the Relationship between Self-Admitted Technical Debt Removals and Technical Debt Measures[J]. Algorithms,2020,13(7):1-16.

[9] HUANG Q,SHIHAB E,XIA X,et al. Identifying self-admitted technical debt in open source projects using text mining[J]. Empirical Software Engineering,2018,23(1):418-451.

[10] MALDONADO E D S,SHIHAB E,TSANTALIS N. Using Natural Language Processing to Automatically Detect Self-Admitted Technical Debt[J]. IEEE Transactions on Software Engineering,2017,43(11):1044-1062.

[11] MALDONADO E D S,SHIHAB E. Detecting and quantifying different types of self-admitted technical Debt[C]// IEEE International Workshop on Managing Technical Debt. IEEE Computer Society,2015:9-15.

[12] WEHAIBI S,SHIHAB E,GUERROUJ L. Examining the Impact of Self-Admitted Technical Debt on Software Quality[C]// Proceedings of the 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering(SANER). IEEE,2016:179-188.

[13] MIKOLOV T,CHEN K,CORRADO G,et al. Efficient Estimation of Word Representations in Vector Space[J]. arXiv:1301.3781,2013.

[14] HOCHREITER S,SCHMIDHUBER J. Long Short-Term Memory[J]. Neural Computation,1997,9(8):1735-1780.

[15] BI L,HU G,RAZA M M,et al. A Gated Recurrent Units (GRU)-Based Model for Early Detection of Soybean Sudden Death Syndrome through Time-Series Satellite Imagery[J]. Remote Sensing,2020,12(21):1-20.

[16] MIAO J,DUAN Y X,ZHANG Y Q,et al. Method for Extracting Event Trigger Words Based on the CNN-BiGRU Model[J]. Computer Engineering,2021,47(9):69-74,83.

[17] CHEN J J,PENG B Z,WU P Z. Malicious Code Detection Method Based on Dynamic Behavior and Machine Learning[J]. Computer Engineering,2021,47(3):166-173.

[18] SCHUSTER M,PALIWAL K K. Bidirectional recurrent neural networks[J]. IEEE Transactions on Signal Processing,1997,45(11):2673-2681.

[19] PENG Z,WEI S,TIAN J,et al. Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification [C]// Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics,2016:207-212.

[20] WANG H,SHI J C,ZHANG Z W. Text semantic relation extraction of LSTM based on attention mechanism[J]. Application Research of Computers,2018,319(5):143-146,166.

[21] REN X X,XING Z C,XIA X,et al. Neural Network-based Detection of Self-Admitted Technical Debt: From Performance to Explainability[J]. ACM Transactions on Software Engineering and Methodology,2019,28(3):1-45.

[22] MAIPRADIT R,TREUDE C,HATA H,et al. Wait for it:identifying "On-Hold" self-admitted technical debt[J]. Empirical Software Engineering,2020,25(5):3770-3798.

[23] XIAO L,CAI Y,KAZMAN R,et al. Identifying and quantifying architectural debt[C]// IEEE/ACM 38th IEEE International Conference on Software Engineering. 2016:488-498.

[24] KIRK B S,PETERSON J W,STOGNER R H,et al. libMesh:a C++ library for parallel adaptive mesh refinement/coarsening simulations[J]. Engineering with Computers,2006,22(3/4):237-254.

XIONG Luo-geng, born in 1996, postgraduate. His main research interests include intelligent software engineering and so on.

ZHENG Shang, born in 1983, Ph.D, associate professor, master's supervisor, is a member of China Computer Federation. His main research interests include intelligent software engineering and data mining.

(责任编辑:何杨)