双粒度轻量级漏洞代码切片方法评估模型

张炳[1,2]，文峥[1,2]，赵宇轩[1]，王苧[1]，任家东[1,2]
（1. 燕山大学信息科学与工程学院，河北 秦皇岛 066004；2. 河北省软件工程重点实验室，河北 秦皇岛 066004）

摘 要：针对现有漏洞代码切片方法评估过程存在的切片信息抽取不完全、模型复杂度高且泛化能力差、评估过程开环无反馈的问题，提出了一种双粒度轻量级漏洞代码切片方法评估模型（VCSE）。针对代码片段，构建了轻量级的 TF-IDF 与 N-gram 融合模型，高效绕过了 OOV 问题，并基于词、字符双粒度提取了代码切片语义及统计特征，设计了高精确率与泛化性能的异质集成分类器，进行漏洞预测分析。实验结果表明，轻量级 VCSE 的评估效果明显优于当前应用广泛的深度学习模型。

关键词：代码切片；漏洞检测；未登录词；轻量级；评估方法
中图分类号：TP309
文献标识码：A
DOI: 10.11959/j.issn.1000−436x.2021196

Dual-granularity lightweight model for vulnerability code slicing method assessment

ZHANG Bing[1,2], WEN Zheng[1,2], ZHAO Yuxuan[1], WANG Ning[1], REN Jiadong[1,2]
1. School of Information Science and Engineering, Yanshan University, Qinhuangdao 066004, China
2. Key Laboratory of Software Engineering of Hebei Province, Qinhuangdao 066004, China

Abstract: Aiming at the problems existing in the assessment of existing vulnerability code slicing method, such as incomplete extraction of slicing information, high model complexity and poor generalization ability, and no feedback in the evaluation process, a dual-granularity lightweight vulnerability code slicing evaluation (VCSE) model was proposed. Aiming at the code snippet, a lightweight fusion model of TF-IDF and N-gram was constructed, which bypassed the OOV problem efficiently, and the semantic and statistical features of code slices were extracted based on the double granularity of words and characters. A heterogeneous integrated classifier with high accuracy and generalization performance was designed for vulnerability prediction and analysis. The experimental results show that the evaluation effect of lightweight VCSE is obviously better than that of the current widely used deep learning model.

Keywords: code slicing, vulnerability prediction, out of vocabulary, lightweight, assessment method

1 引言

漏洞代码切片，旨在分解大规模项目源码为仅含有漏洞相关代码的较小代码切片，并以此消除复杂软件项目中不相关源码对漏洞检测结果的干扰[1-2]。漏洞代码切片新方法的有效性需要经过模型评估来证明。目前，漏洞检测场景中代码切片方法评估模型分为编码模型和机器学习模型两类：编码模型以 Word2Vec、词袋模型、词频−逆文档频率（TF-IDF, term frequency-inverse document frequency）算法[3]为代表；机器学习模型以支持向量机（SVM, support vector machine）、随机森林（RN, random forest）、朴素贝叶斯（NB, naive Bayes）等传统模型为主，循环神经网络（RNN, recurrent

收稿日期：2021−07−23；**修回日期**：2021−09−23
基金项目：国家自然科学基金资助项目（No.61802332, No.61807028, No.61772449）；燕山大学博士基金资助项目（No.BL18012）
Foundation Items: The National Natural Science Foundation of China (No.61802332, No.61807028, No.61772449), The Doctoral Foundation Program of Yanshan University (No.BL18012)

neural network）、长短期记忆网络（LSTM, long short-term memory）、双向长短期记忆网络（BLSTM, bi-directional long short-term memory）等深度学习模型为辅[4]。

然而，现有漏洞代码切片方法评估模型存在以下 3 个主要问题。

1) 代码切片信息抽取不彻底。代码切片具有丰富的语义与结构信息，对这些信息进行全面、高效的提取，是评估代码切片方法有效性的前提。虽然 SVM、RN 等传统机器学习模型收敛速度快、内存占用小，但是文本形式的代码切片需要经过复杂的词嵌入过程才能作为此类模型的输入。并且，在词嵌入后的模型训练过程中，单一的传统机器学习分类器会将不符合或偏离正常模型的切片数据视为噪声，在训练阶段将其忽略，导致漏洞的触发环境、破坏方式、关键数据等代码切片中的隐含信息将在特征缩放、特征编码、模型训练等多个步骤中大量流失。而这些信息恰恰是比较不同切片方法优劣的关键。ELMo[5]、Bert[6]等深度学习方法虽然自身集成了词嵌入过程，但其不仅在训练过程需要大量标注数据与高性能计算的支持，在使用时也需投入一定的模型微调时间。

2) 模型复杂度高且泛化能力差。只有解决词嵌入问题，才能使基于自然语言处理（NLP, natural language processing）的深度学习代码切片评估模型以代码切片序列作为模型输入。而解决未登录词（OOV, out of vocabulary）问题是解决代码词嵌入问题的关键[7]。在实际漏洞检测场景中，存在因代码序列中出现了训练数据集中未出现过的单词，而导致编码失败的现象，即 OOV。由于软件开发者可以在编写程序时自由定义标识符，因此建立包含所有标志符的代码语料库，直接将 NLP 领域的方法迁移至代码切片评估中，会导致词表爆炸问题[6]，代码语料库将会无限大。而通过固定标识符替代不常见词汇，不仅不能缩小词汇表，反而会进一步恶化 OOV 问题[8]。显然，为每个漏洞代码切片方法评估任务人工定制代码语料库的解决办法具有较高的复杂度与极差的泛化能力，将给研究者带来冗余的研究负荷。

3) 模型评估过程开环无反馈。适用于 NLP 的深度学习模型无法可解释地迁移到代码切片方法评估领域。假设代码切片中出现代表文件名称的自定义标识符“GPFilename”。无论是将该文件名分解为“GP、Fil、en、ames”[6]，还是直接将该标识符映射为“Var_n”，都牺牲了原代码的可解释性，丢失了代码切片的语义信息。同时，虽然现有代码切片方法评估模型输出的准确率、马修斯相关系数、F1 值等指标仍可以描述新切片方法对漏洞检测效果的提升程度，但研究者只能基于黑盒的评估模型判断新切片方法是否有效，却无法获取新方法为什么有效、如何改进新方法的相关信息，难以明确人工验证与改进方向。

针对上述问题，本文创新性地提出了双粒度轻量级漏洞代码切片方法评估（VCSE, vulnerability code slicing evaluation）模型，主要包括以下方法。

1) 双粒度提取代码切片特征。通过字符、单词双层次 N-gram[9]模型，捕捉不同粒度、不同窗口大小的单词与字符混合特征，选择性地保留代码切片中相关语义依赖，以统计模型及自然语言处理模型视角获得切片代码的离散表示，避免建立庞大的代码词汇表。最终，以字、词双粒度刻画代码切片特征，从而在特征提取阶段更彻底的保留代码切片中隐含的漏洞信息。

2) 轻量级多模型集成分类。在词嵌入阶段，将 TF-IDF 算法融入 N-gram 模型，以每个 Gram 对应的 TF-IDF 值作为词向量分量，在代码切片向量化的同时绕过 OOV 问题。在嵌入后数据处理阶段，通过构造和组合多个简单机器学习分类器来完成学习任务。利用不同基础分类器的优势和异构性[10]，有效提高模型的预测精度和泛化能力，减少单一分类器带来的误差。最终，研究者可以更快地迭代自身切片方法，或在更短的时间内尝试其他切片技术。

3) 切片检测结果解释反馈。针对分类正确的代码切片，结合其向量表征与对应特征组合，输出该切片中 N-gram 模型捕获到的漏洞标识词组。从而明确被正确分类的代码切片的突出特征，辅助研究者进一步删减代码切片中干扰项，优化切片方法。

实验证明，针对代码切片评估任务，相较于以 Word2Vec[11]、CodeBERT[12]、GraphcodeBERT[13]为代表的已完成超大规模代码预训练的深度学习模型，VCSE 基于改进后的 N-gram 模型，以 TF-IDF 向量表征代码切片，通过轻量级的异质集成模型即可有效判定代码切片是否包含漏洞，并给出判定依据。

2 相关工作

对漏洞代码进行切片能够使代码中间表征更细致地刻画漏洞特征[14]，而任何一种漏洞切片方

法均需要经过恰当的评估模型证明其有效性。Li 等[15]提出了一种只保留内存读写相关代码的切片方法，并通过 BLSTM 神经网络模型评估该切片方法的有效性。μVulDeePecker[16]改进了 Li 等[15]提出的代码切片的提取方式，使代码切片能够应用在多分类任务中，通过融合代码注意力特征的更深层 LSTM 模型进行评估，并以 F1 值为评估指标证明了该切片方法在漏洞多分类任务上的有效性。SySeVR 是一种保留源代码数据与控制依赖的代码切片方法[17]，通过对比 4 种类型漏洞代码切片前后 BGRU 模型输出的 F1 值，证明了 SySeVR 切片方法的有效性。

上述研究明晰了代码切片方法有效性评估过程中的关键是选择恰当的代码中间表征形式与机器学习模型。代码中间表征形式分为度量、文本、树和图 4 种。Chowdhury 等[18]以代码复杂度、耦合和内聚性等度量作为中间表征，学习漏洞代码的度量特征。Mou 等[19]以抽象语法树作为代码的中间表征，基于树结构的卷积网络捕捉源代码的语法结构信息。Zhou 等[20]以扩充的代码属性图作为代码的中间表征，但需借助门图神经网络。

相较于需要专家干预的度量表征、依赖复杂机器学习模型的树和图表征，文本形式的中间代码表征具有直观、不需要代码编译即可获得等良好特性。Hindle 等[21]指出编程语言同样具有自然语言重复性、规律性、可预测性的特点，为采用自然语言处理模型分析代码切片的文本形式中间表征提供了理论依据。Scandariato 等[22]针对源代码将自然语言处理的文本挖掘方法首次引入软件漏洞检测领域。Li 等[15]、μVulDeePecker[16]、SySeVR[17]所用代码切片评估模型均属于自然语言处理模型，并采用词嵌入技术表征文本形式的代码切片。词嵌入技术是融合自然语言处理与漏洞代码切片文本表征的桥梁。OOV 是解决词嵌入问题面临的主要困难之一，且该问题在代码嵌入问题上更加突出。现有词嵌入技术分为上下文无关的词嵌入与基于上下文的词嵌入 2 种类型。Word2Vec、GloVe[23]等上下文无关词嵌入方法，未能考虑单词与上下文之间的语义关系且无法对语料库字典外的单词进行嵌入。ELMo[5]模型与 Bert[6]模型等基于注意力机制的上下文相关的嵌入方法，全面考虑句子中的上下文信息并将其集成到单词的表示中，并采用字节对编码（BPE, byte pair encoding）方法处理 OOV 问题，进一步拆分单词，但是存在模型体量大、复杂度高等问题，仅用于代码预训练效果不佳。

因此，VCSE 选择文本形式作为代码的中间表征，通过 N-gram 模型在考虑代码上文依赖的同时提取代码切片双粒度特征，并以此作为初始语料库。基于 TF-IDF 模型完成词序列向量化过程，从而绕过 OOV 问题。基于代码切片语义及统计特征，设计了高精确率与泛化性能的异质集成分类器，进行漏洞预测分析。

3　模型设计

本节首先介绍 VCSE 整体研究框架，其次介绍 VCSE 主体构成部分：基于 TF-IDF 改进的双粒度 N-gram 模型、基于异质集成学习的轻量级分类器。研究框架中的评估指标、模型解释部分因与实验数据联系紧密，将在第 4 节实验评估中详细展开。

3.1　研究框架

VCSE 总体上分为基于 TF-IDF 改进的双粒度 N-gram 模型和基于异质集成学习的轻量级分类器两部分，如图 1 所示。

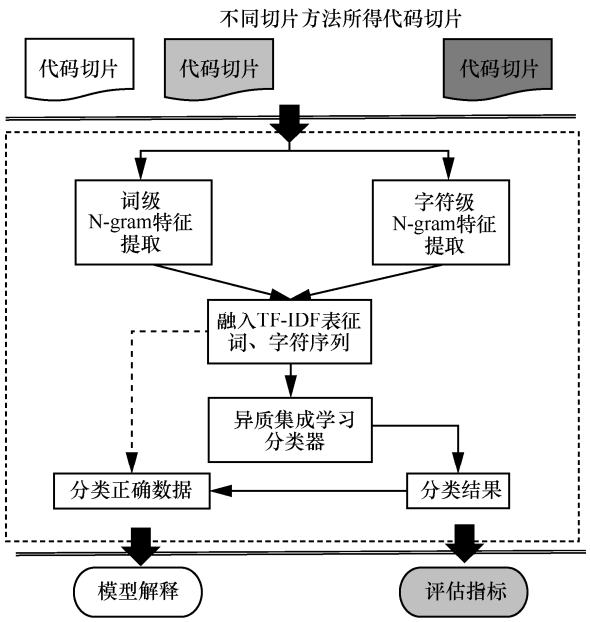

图 1　VCSE 研究框架

1) 基于 TF-IDF 改进的双粒度 N-gram 模型。以 N-gram 捕获的词级、字符级双粒度代码切片语料作为初始特征组合，针对不同代码切片，计算各对应特征分量的 TF-IDF 值，替代词嵌入过程，构建 TF-IDF 向量空间，实现了从代码切片文本到实数矩阵的表示。

2) 基于异质集成学习的轻量级分类器。利用不同分类器在性能上的优势和结构上的差异，选择对漏洞代码切片的检测效果较佳的逻辑回归（LR, logistic regression）、决策树（DT, decision tree）、SVM、RF 作为基础分类器进行两轮学习训练，构建了基于集成学习策略的异质集成分类器。在保证整体模型轻量化的同时，有效提高了模型的准确度与泛化能力。

3.2 基于 TF-IDF 改进的双粒度 N-gram 模型

N-gram 通过长度为 N 的滑动窗口对序列化文本进行分割，形成长度为 N 的特征序列，然后根据设定的频率阈值对特征进行统计和过滤，形成关键的 N-gram 特征列表，即基于文本的特征向量。N-gram 模型旨在寻找长度为 n 的相邻单词的所有组合，它不仅可以提取单个单词，还可以提取单词的序列，捕捉一些其他方法无法提取的潜在特征，适用于代码切片语义信息的离散表示。

由于 N-gram 模型在固定长度的窗口中分割序列化的文本，因此对代码切片进行 N-gram 特征提取时会丢失部分区分能力较强的特征。例如：代码切片中存在函数

memcpy (char *dest, char *src, size n)

从 src 指示的内存地址开始，将 n 个字节复制到 dest 指示的内存地址。若触发缓冲区溢出漏洞。则应满足式(1)～式(5)。

$$\text{allocate(dest)} = [x_0, x_1] \tag{1}$$

$$\text{allocate(src)} = [x_0^{'}, x_1^{'}] \tag{2}$$

$$x_0 - x_1 \geqslant n \tag{3}$$

$$x_0^{'} - x_1^{'} \geqslant n \tag{4}$$

$$x_0 - x_1 \geqslant x_0^{'} - x_1^{'} \tag{5}$$

其中，allocate 为地址分配函数，src 为原内存地址，dest 为目标内存地址，x_0、x_1、$x_0^{'}$、$x_1^{'}$ 为分配到的内存地址，n 为复制字节数。

显然，“memcpy STR len”是识别该缓冲区漏洞的关键特征。然而，如果基于 N 为 1 或 2 的 N-gram 进行特征提取，则提取的特征为“memcpy、STR、len”或“memcpy STR”“STR len”，不能准确表示漏洞特征信息。

此外，随着 N 的增加，提取的特征数量将呈爆炸式增长，不利于后续的预测工作。表 1 显示了以“memcopy (buf str len)”语句为例，当 N 分别为 1、2 和 3 时，分割序列化代码片段的实例。

表 1 序列化代码片段提取到的 N-gram 特征

N	划分结果
1	‘memcopy’, ‘(’, ‘buf’, ‘str’, ‘len’, ‘)’
2	‘memcopy (’, ‘(buf’, ‘buf str’, ‘str len’, ‘len)’
3	‘memcopy (buf’, ‘(buf str’, ‘buf str len’, ‘str len)’

VCSE 采用词级与字符级 2 种窗口提取双粒度的代码切片信息，以保留高信息增益特征。并将 TF-IDF 算法融入 N-gram 模型，从而完成代码切片文本的向量表征。

1) 高信息增益特征保留

综合考虑不同窗口提取的特征和单词形态对预测结果的影响，基于代码切片提取不同粒度和不同窗口大小的词与字符级别的 N-gram 特征。词级 N-gram 模型在提取特征时以词为单位，并忽略“(”“)”“;”等无意义定界符特征。字符特征是构成软件代码的最基本元素，字符级 N-gram 模型提取单词中的标识符和字母作为补充特征，深入研究代码切片中词的形态构成。综合上述 2 种提取方式，最终通过字符、词 2 种粒度获取与漏洞紧密相关的文本特征，从而保留高信息增益特征，增强稳健性。

2) OOV 绕过的代码切片表征方法

为获得文本形式的代码切片中间表征，并解决双粒度的 N-gram 模型存在的维数灾难问题，将 TF-IDF 算法集成到 N-gram 模型，筛选出具有更佳分类能力的特征，以特征项 TF-IDF 值的计算替代了词嵌入过程，从而避免了词嵌入过程中存在的 OOV 问题。

针对代码切片集合提取不同窗口大小的 N-gram 特征，得到词级特征集与字符级特征集分别为

$$D = \{d_1, d_2, \cdots, d_n\} \tag{6}$$

$$S^k = \{t_1, t_2, \cdots, t_q\}(k = 1, 2) \tag{7}$$

其中，D 为代码切片数据集，$d_i(1 \leqslant i \leqslant n)$ 为代码切片，$t_j(1 \leqslant j \leqslant q)$ 为提取到的特征，S^1 为词级特征集，S^2 为字符级特征集。

在少数代码切片中，频繁出现的特征项显然具有更好的分类能力。为赋予该类特征更高的权值，基于 TF-IDF 算法计算不同特征集代码切片的特征项频率与逆文档频率，最终计算得到每个特征项的

TF-IDF 值。计算方法如式(8)～式(10)所示。

$$\mathrm{tf}_{i,j}=\frac{N_{i,j}}{\sum_{h=1}^{q}N_{h,j}} \tag{8}$$

$$\mathrm{idf}_i=\log\frac{\|D\|}{1+\|D_{t_i}\|} \tag{9}$$

$$\mathrm{tfidf}_{t_j}=\mathrm{tf}_{t_j}\mathrm{idf}_{t_j} \tag{10}$$

其中，$\mathrm{tf}_{i,j}$、idf_i 分别为 t_i 的特征项频率与逆文档频率，$N_{i,j}$ 为特征项 i 出现在 d_j 中次数，$N_{h,j}$ 为 d_j 中第 h 个特征项出现的次数，$\mathrm{tfidf}_{i,j}$ 为第 i 个代码切片的中间表征中第 j 个特征项的值。

为了选择出分类能力强的特征项，分别根据 TF-IDF 值的大小对词、字符级特征集进行排序。并分别选取排名靠前的特征构成词、字符特征子集。最终，拼接两子集构成 VCSE 分类特征。

$$S_{\mathrm{sub}}^{k}=\{w_1^k,w_2^k,\cdots,w_n^k\} \tag{11}$$

$$S_{\mathrm{sub}}=\bigcup_{k=1}^{2}S_{\mathrm{sub}}^{k} \tag{12}$$

其中，w_i^k 表示第 i 个词或字符级特征对应的 TF-IDF 值，S_{sub}^{k} 表示代码切片的词或字符级特征表示，S_{sub} 表示拼接后的 VCSE 分类特征。

最终代码切片的文本形式中间表征如表 2 所示。

表 2　代码切片的文本形式中间表征示例

D	t_1	t_2	…	t_m
d_1	$\mathrm{tfidf}_{1,1}$	$\mathrm{tfidf}_{1,2}$	…	$\mathrm{tfidf}_{1,m}$
d_2	$\mathrm{tfidf}_{2,1}$	$\mathrm{tfidf}_{2,2}$	…	$\mathrm{tfidf}_{2,m}$
⋮	⋮	⋮	⋱	⋮
d_n	$\mathrm{tfidf}_{n,1}$	$\mathrm{tfidf}_{n,1}$	…	$\mathrm{tfidf}_{n,m}$

3.3　基于异质集成学习的轻量级分类器

BLSTM、Bert 等深度学习算法往往针对复杂的语意推断与意图识别任务而设计，将其应用于代码切片方法的评估任务固然能获得较高的准确率，其高复杂度却阻碍了切片方法的快速迭代改进，且经过 BPE 算法分词后得到的特征严重丢失可解释性。而 lightgbm、catboost 等基于基础分类算法的改进算法虽然优化了原有算法的部分缺点，但仍无法完全克服原有单一基础分类器在某些方面存在的不足，如准确率较高但时间较长、误报率较低但漏报率较高。模型的性能与选择的算法和数据的特征均密切相关。对于不同代码切片数据集，不同分类器的性能往往是不同的。为了得到各方面稳定、高性能的模型，采用异质集成策略学习基础分类器的输出结果，并利用各基础分类器结构和性能优势的多样性，构建稳定、泛化能力强的预测模型。

异质集成学习预测模型的性能主要取决于 2 个因素，一个是基础分类器的多样性，另一个是基础分类器的预测精度。首先，满足多样性，选择 LR、DT、RF、NB 和 SVM 算法等广泛应用于对比评估任务、具有较强稳定性和适应性、在不同的指标上各有优势的分类算法作为基础分类器。其次，基础分类器的预测精度越高、结构差异越大，最终的集成效果会越好。因此，以预测精度为过滤指标进行验证测试，选择 SVM、LR 两类线性模型和 DT、RF 两类非线性模型作为基本分类器。

异质集成学习策略需要两轮学习才能得到最终的分类器。针对同一代码切片数据集，首先训练不同的基本分类器。为了减少弱分类器的训练误差，提高模型的泛化性能和预测精度，将每个基本分类器的输出结果作为第二轮训练的输入，进而得到最终分类器。由于代码切片方法的评估任务只需检测输入代码切片中是否含有相应漏洞，因此最终分类器只需完成二元分类任务。故在第二轮训练中使用简单、低复杂度的 LR 作为分类器，具体如图 2 所示。

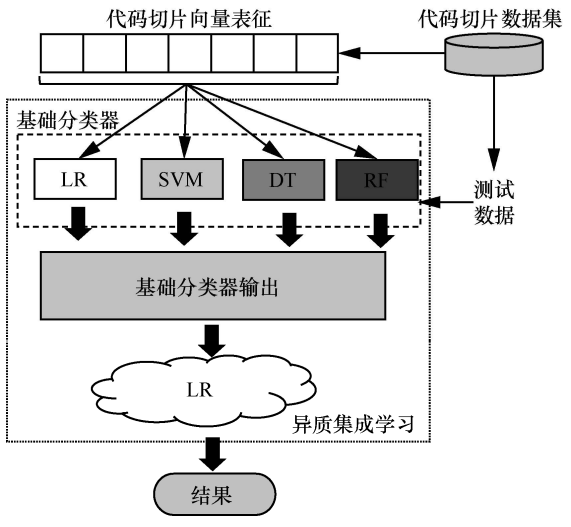

图 2　异质集成学习方法预测过程

首先，使用训练数据集来构建 LR、SVM、DT、RF 这 4 个不同的基础分类器，从而输出第一轮的预测结果。

$$D=\{x_i \in R^d, y_i \in \{0,1\}\} \tag{13}$$

其中，x_i 为切片特征向量，y_i 为 1 或 0 分别代表该代码切片是否有漏洞。

然后，根据不同结构基础分类器的输出和代码切片的实际标签值，构造最终分类器的输入数据。最终分类器 LR 通过梯度下降算法输出代码切片检测结果。最小化损失函数为

$$T=\frac{1}{1+\mathrm{e}^{-\omega^T x}} \tag{14}$$

$$J(\omega)=-|\sum_{i=1}^{n} y_i \log T+(1-y_i)\log(1-T)| \tag{15}$$

其中，ω 为待求参数，T 为 sigmoid 函数。

最终，VCSE 模型训练过程如算法 1 所示。

算法 1 VCSE 模型训练算法

输入 漏洞代码切片数据集 D

输出 双粒度特征集 F、切片分类检测结果 R

1) 针对 D 中任一代码切片 d_i，进行小写化与驼峰化处理。

2) 划分窗口，获取大小为 N_1 的字符子序列与大小为 N_2 的词特征子序列。

3) 遍历 D，并重复步骤 1)和步骤 2)。

4) 计算每个子序列 TF-IDF 值，计算方法如式(8)～式(10)所示。

5) 根据 TF-IDF 值的大小对词、字符级特征集 S^1、S^2 进行排序。

6) 水平拼接排序后 S^1 前 N_3 个特征与及 S^2 前 N_4 个特征，共同构成 VCSE 分类特征 S_{sub}。

7) 针对任一漏洞切片 d_i，构建代码切片向量的文本形式中间表征向量 $\boldsymbol{V}_i$。若切片中包含 S_{sub} 中的特征分量，即将 $\boldsymbol{V}_i$ 对应位置设为该特征分量在 d_i 中的 TF-IDF 值。反之，记为 0。

8) 遍历 D，并重复步骤 7)，构建训练数据集 T。

9) 以 T 作为异质集成分类器输入，训练 VCSE 模型。

4 实验评估

本节首先阐述 VCSE 所用实验数据集及评估指标。然后最优化 VCSE 模型参数，确定 VCSE 模型。最后一方面以缓冲区溢出切片数据集为例，将 VCSE、不同基础分类器、BLSTM 进行对比，以证明 VCSE 优于上述模型；另一方面以资源管理、数组使用、算术表达 3 个切片数据集为例，将 VCSE、TextCNN 与原论文评估模型对比，从而证明相较适用广泛的深度学习模型，轻量级的 VCSE 已足够满足代码切片评估任务需求。

本文实验的实验环境配置如下：CPU 为 Intel Xeon，内存为 12 GB 的 Colab 服务器，操作系统版本为 Ubuntu 18.04.5 LTS。

4.1 数据集及评估指标

代码切片数据集即依据代码切片方法处理后的漏洞代码数据集。VCSE 实验所用代码切片数据以美国国家漏洞库（NVD, national vulnerability database）与软件质量保证参考数据集（SARD, software assurance reference dataset）中部分 C/C++程序漏洞代码为基础，经由 Li 等[15]与 SySeVR [17]中提出的切片算法处理后得到。具体为来源于 NVD 数据集的 1 591 个开源 C/C++程序以及取自 SARD 数据集的 14 000 个 C/C++程序。

依据切片算法对漏洞代码切片后可以得到包含缓存区溢出、资源管理、数组使用、算术表达 4 种不同语法特征的代码切片数据集。随机选择其中 80%用于 VCSE 模型训练，20%用于测试。代码切片数据集信息如表 3 所示。

表 3 代码切片数据集信息

类型	含有漏洞的代码切片数量/个	不含漏洞的代码切片数量/个	漏洞代码切片占比
缓存区溢出	10 400	39 753	26.3%
资源管理	7 285	21 885	33.3%
数组使用	10 926	42 229	25.9%
算术表达	3 475	22 154	15.7%

由 4 种不同类型漏洞代码切片数据集中含有漏洞的代码切片占比可知，代码切片数据集存在不平衡问题，即含有漏洞的代码切片数量远少于不含漏洞的代码切片数量。针对上述问题，VCSE 以误报率（FPR, false positive rate）、漏报率（FNR, false negative rate）、召回率（TPR, true positive rate）、准确率（P, precision）以及 F1 值（F1, F1-measure）综合评估模型。

$$\text{FPR}=\frac{\text{FP}}{\text{FP+TN}} \tag{16}$$

$$\text{FNR}=\frac{\text{FN}}{\text{TP+FN}} \tag{17}$$

$$\text{TPR}=\frac{\text{TP}}{\text{TP+FN}} \tag{18}$$

$$P=\frac{TP}{TP+FP} \tag{19}$$

$$F1=\frac{2TP}{2TP+FP+FN} \tag{20}$$

其中，TP 表示含有漏洞的代码切片被分类器检测成功的数量，FP 表示不含漏洞的代码切片被分类器误检测的数量，TN 表示为不含漏洞的代码切片被分类器正确识别的数量，FN 表示为含有漏洞的代码切片未被分类器识别的数量。

4.2　VCSE 模型参数最优化

不同 N 值的 N-gram 模型在语义特征提取时将采用不同大小的窗口。而 N-gram 模型窗口的大小将直接影响 VCSE 特征提取阶段可以获得的约束性信息。虽然随着 N 值的增加，模型获得的特征更全面，但是也会造成模型复杂度的指数级增加。因此，合适的 N 值应综合权衡模型性能及复杂度。

通过受试者工作特征（ROC, receiver operating characteristic）曲线，分别对词级与字符级 N-gram 模型 N 取不同值时的特征提取效果进行检验，结果如图 3 和图 4 所示。

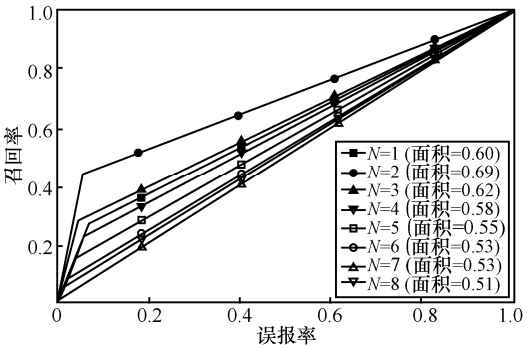

图 3　词级特征提取不同 N 值的结果

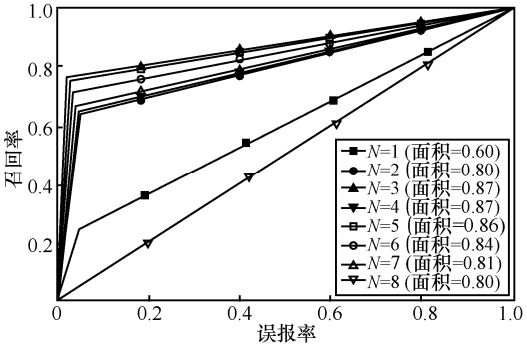

图 4　字符级特征提取不同 N 值的结果

由图 3 和图 4 可知，针对词级的 N-gram 特征提取，N=2 时效果最好，其次是 1 和 3。针对字符级的 N-gram 特征提取，N 取 3～7 时效果最好。综合考虑时间消耗和性能，最终选取词级的 N 值为 1、2、3，字符级的 N 值为 3、4、5。

除 VCSE 采用的 N-gram 模型外，Doc2vec 模型、词袋模型等均为文本信息常用量化表示方法。以准确率、召回率、F1 值为指标，以 Doc2vec 模型、词袋模型替换 N-gram 模型进行对比，结果如图 5 所示。

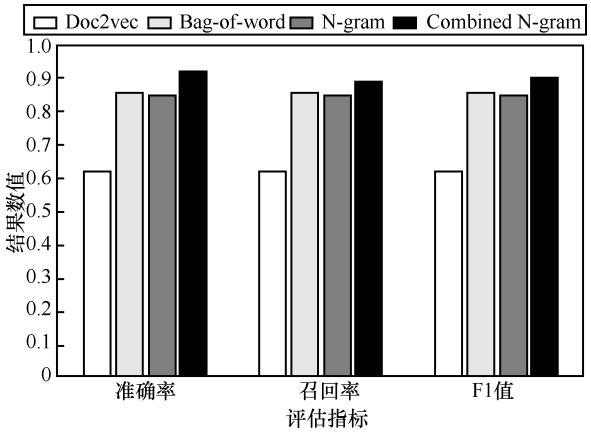

图 5　不同特征提取模型的对比结果

图 5 结果表明，VCSE 采用 N-gram 模型的准确率、召回率和 F1 值分别为 92%、89%和 90%，与其他 3 种模型相比有显著提高。

4.3　VCSE 评估效果

本节通过以下两方面来验证 VCSE 有效性。

1) 验证 VCSE 抽取的特征组合 S_{sub} 对代码切片具有更好的表征效果。

2) 验证 VCSE 对不同代码切片方法具有普适性。

首先，基于缓存区溢出漏洞的代码切片数据集，以 VCSE 提取的代码切片对应特征 S_{sub} 作为不同基础分类器的输入，并将分类效果与基于代码嵌入及深度学习的 VulDeePecker[15] 工具（Word2Vec+BLSTM 分类器）进行对比。对比结果如表 4 所示。

表 4　缓存区溢出类型下的漏洞检测表现

分类器	FPR	FNR	TPR	P	F1
Word2Vec+BLSTM	2.9%	18.0%	82.0%	91.7%	86.6%
S_{sub}+SVM	3.1%	18.4%	81.6%	90.5%	85.8%
S_{sub}+RF	2.4%	14.8%	85.2%	92.7%	88.8%
S_{sub}+TextCNN	2.3%	20.2%	79.8%	92.5%	85.7%
VCSE	2.7%	11.4%	88.6%	92.1%	90.3%

表 4 中，除最简单的 SVM 分类器检测效果与 VulDeePecker[15]中原分类器检测效果相近外，其余复杂度较高的分类器对代码切片的检测效果均优于原分类器。初步证明，VCSE 抽取的特征组合 S_{sub} 对代码切片具有更佳表征能力，能够有效弥补基础分类算法自身分类能力的不足。

由于 SVM、RF 均集成于 VCSE 使用的异质分类器中，故在仅保留 VulDeePecker[15]原分类器（表 5 中 Word2Vec+BLSTM 分类器）与 TextCNN 的情况下，基于资源管理漏洞的代码切片数据集，二次验证 VCSE 抽取的特征组合对代码切片的表征能力，结果如表 5 所示。

表 5 资源管理类型下的漏洞检测表现

分类器	FPR	FNR	TPR	P	F1
Word2Vec+BLSTM	2.8%	4.7%	95.3%	94.6%	95.0%
S_{sub}+TextCNN	1.8%	7.6%	92.3%	96.2%	94.2%
VCSE	1.2%	2.6%	97.4%	97.7%	97.5%

由表 5 可知，VCSE 在精确率与 F1 值上较于 VulDeePecker[15]分别提升 3.1%、2.5%。进一步证明了 VCSE 抽取的特征组合 S_{sub} 对代码切片具有更好的表征效果。

为验证 VCSE 对不同代码切片方法具有普适性，基于 SySeVR[17]中保留源代码数据与控制依赖的切片方法得到的漏洞切片数据集，对比 VCSE 提出的轻量级异质集成分类器、SySeVR[17]原评估模型（Word2Vec+BLSTM 分类器）、TextCNN 算法的漏洞代码切片检测表现。结果分别如表 6 和表 7 所示。

表 6 数组使用类型下的漏洞检测表现

分类器	FPR	FNR	TPR	P	F1
Word2Vec+BLSTM	3.8%	17.1%	92.7%	88.3%	85.5%
S_{sub}+TextCNN	4.0%	12.9%	87.2%	88.5%	87.8%
VCSE	4.6%	9.8%	90.1%	87.2%	88.7%

表 7 算术表达类型下的漏洞检测表现

分类器	FPR	FNR	TPR	P	F1
Word2Vec+BLSTM	1.5%	18.3%	96.9%	87.9%	84.7%
S_{sub}+TextCNN	0.9%	53.1%	47.0%	90.8%	61.9%
VCSE	1.2%	12.7%	87.3%	93.0%	90.1%

表 6 和表 7 中，在算术表达类型下精确率与 F1 值上较于 SySeVR[17]分别提升 5.1%、5.4%。虽然相较于 SySeVR[17]，VCSE 在数组使用类型下精确率下降 1.1%，但 F1 值提升了 3.2%。F1 值是在综合考量精确率与召回率的基础上得到的，更能全面地评估模型，且 F1 值提升幅度大于精确率下降幅度。故 VCSE 对不同代码切片方法具有普适性。

在达到上述较优效果的同时，VCSE 模型避免了代码词汇表的建立，且模型使用的双粒度特征组合 S_{sub} 维数仅为 1 650。传统的基于代码词嵌入的深度学习模型[8]，若使用部分特征将代码词汇表中词汇数目控制在 75 000 以内，则词汇表仅能包含真实使用场景中 20%的词汇，且仅有 17%的词汇出现频率为 10 次以上，25%的词汇仅能出现一次。同时，VCSE 使用双粒度 N-gram 模型提取的 1 650 维特征组合 S_{sub} 具备可解释性。例如，基于缓存区溢出类型的漏洞代码切片提取的 S_{sub} 中，含有“malloc”“buffer”“strlen”等内存操作关键字。其部分特征名称如图 6 所示。

“buf strlen”, “buf strlen buf”, “char malloc”, “char array size”, “free”, “fun2 strlen”, “malloc 100 sizeof”, “memcpy”, “memset”, “new”, “null”, “null data”, “sizeof char”, “sizeof wchart”, “strcpy”, “strcpy fun2”, “strlen” ···

图 6 VCSE 抽取的特征组合中部分特征名称

5 结束语

本文提出了一种未登录词绕过的双粒度轻量级漏洞代码切片方法评估模型，融合 N-gram 模型与 TF-IDF 算法，在绕过 OOV 问题的同时更好地表征代码切片。构建轻量级的异质集成学习分类器替代高复杂度的深度学习分类器，减轻代码切片研究人员冗余研究负担。实验证明，针对漏洞代码方法评估领域，VCSE 在不使用 Word2Vec 引入高维向量与深度学习模型的前提下，性能及训练成本均优于原有切片方法评估过程使用的深度学习模型。未来工作将考虑基于 VCSE 抽取的特征组合 S_{sub} 作为漏洞 Wordpieces 扩充到 Bert 等深度学习模型词汇语料库，修改 Bert 模型部分目录下词汇语料库文件，帮助迁移 Bert 等深度学习模型到漏洞检测领域。

参考文献：

[1] LIN G J, WEN S, HAN Q L, et al. Software vulnerability detection using deep neural networks: a survey[J]. Proceedings of the IEEE,

2020, 108(10): 1825-1848.

[2] 李珍, 邹德清, 王泽丽, 等. 面向源代码的软件漏洞静态检测综述[J]. 网络与信息安全学报, 2019, 5(1): 1-14.
LI Z, ZOU D Q, WANG Z L, et al. Survey on static software vulnerability detection for source code[J]. Chinese Journal of Network and Information Security, 2019, 5(1): 1-14.

[3] RAMOS U J. Using tf-idf to determine word relevance in document queries[J]. Proceedings of the First Instructional Conference on Machine Learning, 2003, 242: 133-142.

[4] 李韵, 黄辰林, 王中锋, 等. 基于机器学习的软件漏洞挖掘方法综述[J]. 软件学报, 2020, 31(7): 2040-2061.
LI Y, HUANG C L, WANG Z F, et al. Survey of software vulnerability mining methods based on machine learning[J]. Journal of Software, 2020, 31(7): 2040-2061.

[5] PETERS M E, NEUMANN M, IYYER M, et al. Deep contextualized word representations[J]. arXiv Preprint, arXiv:1802.05365, 2018.

[6] DEVLIN J, CHANG M W, LEE K, et al. Bert: pre-training of deep bidirectional transformers for language understanding[J]. arXiv Preprint, arXiv:1810.04805, 2018.

[7] BURATTI L, PUJAR S, BORNEA M, et al. Exploring software naturalness through neural language models[J]. arXiv Preprint, arXiv:2006.12641, 2020.

[8] KARAMPATSIS R M, BABII H, ROBBES R, et al. Big code != big vocabulary: open-vocabulary models for source code[C]//Proceedings of Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering. New York: ACM Press, 2020: 1073-1085.

[9] BROWN P F, DELLA PIETRA V J, SOUZA P V, et al. Class-based N-gram models of natural language[J]. Computational Linguistics, 1992, 18(4): 467-479.

[10] DIETTERICH T G. Ensemble learning[J]. The Handbook of Brain Theory and Neural Networks, 2002, 2(1): 110-125.

[11] MIKOLOV T, CHEN K, CORRADO G, et al. Efficient estimation of word representations in vector space[J]. arXiv Preprint, arXiv:1301.3781, 2013.

[12] FENG Z Y, GUO D Y, TANG D Y, et al. Codebert: a pre-trained model for programming and natural languages[J]. arXiv Preprint, arXiv:2002.08155, 2020.

[13] GUO D Y, REN S, LU S, et al. GraphCodeBERT: pre-training code representations with data flow[J]. arXiv Preprint, arXiv:2009.08366, 2020.

[14] SALIMI S, EBRAHIMZADEH M, KHARRAZI M. Improving real-world vulnerability characterization with vulnerable slices[C]//Proceedings of Proceedings of the 16th ACM International Conference on Predictive Models and Data Analytics in Software Engineering. New York: ACM Press, 2020: 11-20.

[15] LI Z, ZOU D Q, XU S H, et al. VulDeePecker: a deep learning-based system for vulnerability detection[J]. arXiv Preprint, arXiv:1801.01681, 2018.

[16] ZOU D Q, WANG S J, XU S H, et al. μVulDeePecker: a deep learning-based system for multiclass vulnerability detection[J]. IEEE Transactions on Dependable and Secure Computing, 2021, 18(5): 2224-2236.

[17]] LI Z, ZOU D Q, XU S H, et al. S_ySeVR: a framework for using deep learning to detect software vulnerabilities[J]. IEEE Transactions on Dependable and Secure Computing, 2021, PP(99): 1.

[18] CHOWDHURY I, ZULKERNINE M. Using complexity, coupling, and cohesion metrics as early indicators of vulnerabilities[J]. Journal of Systems Architecture, 2011, 57(3): 294-313.

[19] MOU L L, LI G, ZHANG L, et al. Convolutional neural networks over tree structures for programming language processing[J]. arXiv Preprint, arXiv:1409.5718, 2014.

[20] ZHOU Y, LIU S, SIOW J, et al. Devign: effective vulnerability identification by learning comprehensive program semantics via graph neural networks[J]. arXiv Preprint, arXiv:1909.03496, 2019.

[21] HINDLE A, BARR E T, GABEL M, et al. On the naturalness of software[J]. Communications of the ACM, 2016, 59(5): 122-131.

[22] SCANDARIATO R, WALDEN J, HOVSEPYAN A, et al. Predicting vulnerable software components via text mining[J]. IEEE Transactions on Software Engineering, 2014, 40(10): 993-1006.

[23] PENNINGTON J, SOCHER R, MANNING C. Glove: global vectors for word representation[C]//Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Stroudsburg: Association for Computational Linguistics, 2014: 1532-1543.

[作者简介]

张炳（1989− ），男，湖北黄冈人，博士，燕山大学副教授、硕士生导师，主要研究方向为数据挖掘、机器学习、软件安全。

文峥（1998− ），男，河北保定人，燕山大学硕士生，主要研究方向为软件安全。

赵宇轩（1997− ），男，河北秦皇岛人，燕山大学硕士生，主要研究方向为文本挖掘、软件安全。

王苧（1994− ），女，山西阳泉人，燕山大学硕士生，主要研究方向为软件安全。

任家东（1967− ），男，黑龙江齐齐哈尔人，博士，燕山大学教授、博士生导师，主要研究方向为时态数据建模、软件安全。