文章编号：1003－1251(2024)01－0061－08

一种基于 FastText 的恶意代码家族分类方法

张宇迪，冯永新，赵运弢

（沈阳理工大学 信息科学与工程学院，沈阳 110159）

摘　要：传统的恶意代码家族分类方法主要通过代码家族浅层关联特征的统计分析达到分类和识别的目的。随着恶意代码加壳、混淆、多态技术的发展，传统方法的局限性逐渐显现，但恶意代码需调用 API 函数达成恶意目的始终是其不变的行为特征。基于 embedding、word2vec 模型的传统方法缺乏对低频 API 函数的特征提取能力，在表征 API 序列局部顺序特征时易产生映射失真，存在词典外 API 行为扩展、推理能力弱等导致分类准确率下降的不足。由此，引入负采样优化的 FastText 框架以加强对 API 序列映射的准确度，提出一种基于 FastText 框架下的恶意代码家族分类方法。利用 FastText 框架实现代码样本 API 序列的多维向量转换和精准表达，结合一维卷积及长短时记忆(LSTM)网络进一步提取 API 行为局部特征。实验结果表明，该模型的性能相较于传统的 embedding 方法和 word2vec 框架性能更优，准确率可达 99% 以上。

关　键　词：FastText；恶意代码家族分类；长短时记忆网络

中图分类号：TN915.08　　**文献标志码：**A　　**DOI：**10.3969/j.issn.1003－1251.2024.01.010

A Classification Method of Malicious Code Family Based on FastText

ZHNAG Yudi，FENG Yongxin，ZHAO Yuntao

（Shenyang Ligong University，Shenyang 110159，China）

Abstract：The traditional classification method of malicious code families is mainly used to achieve the purpose of classification and recognition through statistical analysis of shallow association features of code families. With the development of malicious code shelling，obfuscation，and polymorphism techniques，the limitations of the traditional methods are gradually emerging. However，malicious code needs API functions to achieve malicious purposes. Traditional methods based on embedding and word2vec models are unable to extract features from low-frequency API functions，and are prone to mapping distortion when characterizing local sequential features of API sequences. These methods also have shortcomings such as extended API behavior outside the dictionary and weak reasoning ability，which can lead to a decrease in classification accuracy. Therefore，a negative sampling optimized FastText framework is introduced to enhance the accuracy of API sequence mapping，and a malicious code family classification method based on the FastText framework is proposed. The FastText framework is utilized to achieve multidimensional vector transformation and precise expression of code sample API sequences，and one-dimensional convolution and long short term memory(LSTM)networks are combined to further extract local features of API behavior. The experimental results show that the performance of this model is superior to traditional embedding

收稿日期：2023－01－23

基金项目：国家自然科学基金项目(61971291)

作者简介：张宇迪(1997—)，男，硕士研究生；冯永新(1974—)，通信作者，女，教授，博士，研究方向为网络安全、移动无线网络技术。

methods and the Word2vec framework, with an accuracy rate of over 99%.

Key words: FastText; malicious code family classification; long short term memory networks

随着网络普及率越来越高,网络信息交互越来越多,网络攻击者会编写影响范围更广、更具隐蔽性的恶意代码以达到破坏网络、窃取信息等非法目的[1],该类恶意代码大多使用变形或多态技术躲避计算机杀毒软件的检测。恶意代码家族是指具有相似功能、相同来源和不同演进程度的恶意代码集合[2],其分类研究可以视为对不同恶意代码的相似性、关联性判断,有助于研究人员发现同类型的变异代码并快速了解恶意代码感染策略、威胁级别[3]等信息,进而采取针对性措施防止恶意代码对网络的破坏。

目前已存在多种恶意软件和快速更新的反检测技术[4],恶意代码的检测可根据是否实际运行代码样本分为静态检测和动态检测。与动态检测技术相比,静态检测技术具有能耗低、风险低、速度快和能识别非触发性、潜伏性的恶意行为等优点。在针对 API 序列的静态检测技术中,人工神经网络与词向量的结合显示出独特优势[5]。例如,使用长短时记忆网络(LSTM)和循环神经网络(RNN)提取相关特征的模型和使用 Apigraph 技术[6]加强训练的 Android 和 Windows 恶意代码检测系统。王博等[7]将二进制序列分割成 RGB 三通道,采用 VGGNet 模型针对恶意程序变种的代码复用进行分类检测,该模型的准确率可达 96.16%。杨宇夏等[8]将 N-grams 特征与恶意代码灰度图相结合,用于解决样本大小不一的问题并从文本和灰度图两个维度同时提取特征,实验结果表明该文提出的融合特征模型准确率可达 98%。王栋等[9]基于 VGG 模型构建了一维卷积分类网络模型 ID-CNN-IMIR,相较于其他深度学习模型具有更好的性能,准确率可达 98.94%。静态分析的缺点是容易受到混淆的影响,可直接分析原始程序代码的语法和语义,进而提取到更深层次的特征。

在针对 API 序列的静态检测中,常用 word2vec 模型将 API 序列映射为词向量。word2vec 首先生成词典,再将此词典中的所有 API 函数映射为词向量。所生成的词向量仅是样本语义到向量的简单映射,局限于 API 函数的顺序特征且依赖词典,缺乏对词典外的 API 函数表征能力。恶意代码随着混淆技术的发展,执行程序的复杂程度大大提升,如何进一步表征 API 序列的内在特征变得尤为重要。

1 面向恶意代码检测的 FastText 框架分析与设计

在恶意代码家族分类中,受文献[10]启发,本文对收集的 API 数据进行向量化。由于卷积神经网络(CNN)和长短时记忆(LSTM)网络在特征提取方面效果突出,二者相结合的深度学习模型对局部特征的提取更加完备,因此采用基于词嵌入的 CNN + LSTM 作为主要网络模型。针对 word2vec 只是对样本语义的简单映射[11],对低频和相似 API 函数映射不准确的不足,采用基于 N-gram 特征的 FastText 框架结合负采样技术进行向量化,使用 CNN + LSTM 的深度学习模型进行训练,称为 FastText + CNN + LSTM 模型。FastText 模型是 Facebook 公司一个开源的 NLP 工具[12],主要应用于文本序列的向量化和快速分类。其词向量的训练既可以有监督也可以无监督,训练完成后的词向量可进一步应用于其他模型或特征选择等任务。FastText 模型本质上是对 word2vec 中词袋模型(CBOW)的改进,并配合一个预训练好的线性分类器[13]使用,可采用逻辑回归模型、支持向量机等。本文首先通过实验研究 FastText 框架中每个 API 函数被分解的最小字段长度 n 取值对模型整体性能的影响,确定最佳的 n 值,并与传统的 embedding + CNN + LSTM 模型和 word2vec 模型进行对比,最后通过十折交叉实验对模型性能进行验证。

1.1 基础框架

FastText 的组成主要分为输入层、隐藏层和输出层三部分。通过输入层将样本文本转为对应的 N-gram 特征向量[14],在隐藏层中进行输入向量的叠加平均,输出层采用归一化指数函数[15](Softmax)预测结果。本文使用 FastText 输入层对 API 调用序列进行向量化,该层主要结构如图 1 所示。

FastText 模型的输入层由两部分向量组成[16],分别是分词词典中对每个 API 函数的词嵌入(embedding)向量$\{w_{j,1}, w_{j,2}, \cdots, w_{j,n}\}$和经过 N-gram 特征取词后的 embedding 向量$\{w_{t,1}, w_{t,2}, \cdots, w_{t,n}\}$,其中 j 和 t 分别为同个 API 函数两个分向量

的编号,n 为两个分向量的维度,通过两个向量相加操作得到模型需要的词向量$\{w_{j,1}+w_{t,1}, w_{j,2}+w_{t,2}, \cdots, w_{j,n}+w_{t,n}\}$。由上述过程可以看出,FastText 模型对 API 序列处理具有 N-gram 特征。

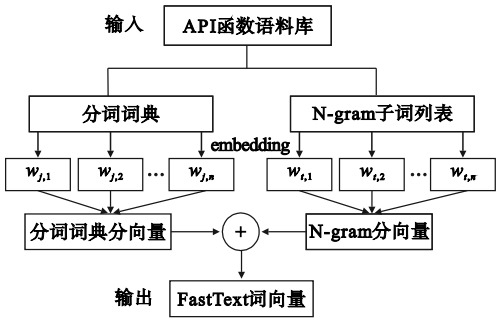

图 1　FastText 输入层结构图

Fig. 1　FastText input layer structure diagram

为达到分类的效果,FastText 模型的目标是将极大似然转换为对数似然,并最小化该目标函数。目标函数的计算方法为

$$LL = -\frac{1}{k}\sum_{i=1}^{k} Y_i \log(f(\bar{\boldsymbol{B}}\boldsymbol{X}_i)) \qquad (1)$$

式中:Y_i 表示第 i 个恶意样本对应的标签值;k 为负样本数;f 表示使用 Softmax 函数预测类别;$\bar{\boldsymbol{B}}$ 是权重矩阵[17];$\boldsymbol{X}_i$ 为第 i 个样本特征向量。$\boldsymbol{X}_i$ 表达式为

$$\boldsymbol{X}_i = \bar{\boldsymbol{A}}[\text{func}(v_1, v_2, \cdots, v_{n-1}, v_n)] \qquad (2)$$

式中:func 是叠加平均函数,计算输入 API 函数序列的特征向量;$\bar{\boldsymbol{A}}$ 是权重矩阵;$(v_1, v_2, \cdots, v_{n-1}, v_n)$是输入样本中的 N-gram 取词后的词向量。

1.2　面向 API 序列的负采样 FastText 模型优化

FastText 根据前后 API 序列预测当前位置 API 函数出现概率,其训练需要通过输入样本不断调整神经元的权重提高对目标预测的准确率。当文本较复杂时,需要消耗较大的算力调整模型,并且向量化的 API 函数数量众多,会导致对低频 API 的映射精确度下降[18]。因此,本文引入负采样技术对 FastText 模型进行优化。负采样技术通过引入负例,每次只更新与目前待训练 API 有关的部分神经元,降低了运算量并改善了输出词向量的质量。

负采样优化首先确定中心 API 函数并设置窗口长度,以中心 API 函数的前后序列作为正样本,采用词汇表中其余 API 函数为负样本,通过二元逻辑回归求解每个 API 函数对应的待训练参数[19]和中心 API 对应前后序列每个 API 的词向量[20]。将正负样本采用二元回归模型进行分类,根据链式法则和激活函数以梯度上升法对二元回归模型的可训练参数进行优化。基于负采样优化的 FastText 模型及其训练流程如图 2 所示。

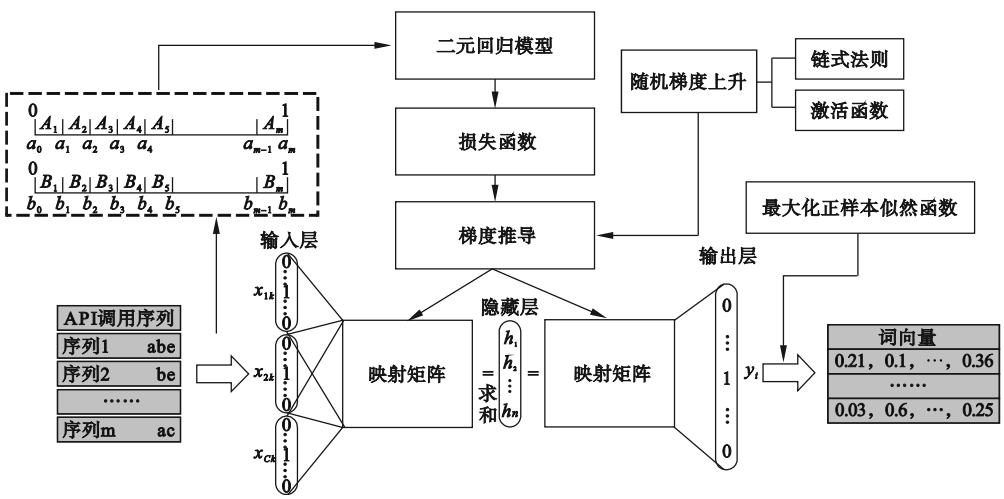

图 2　FastText 模型及其优化流程示意图

Fig. 2　FastText model and its optimization process diagram

负采样优化的 FastText 首先从输入 API 序列(长度为 m)中随机抽取一个样本$\{x_{1k}, x_{2k}, \cdots, x_{Ck}\}$,其中 C 为样本维度数,经过 FastText 模型处理生成一个词汇表,以中心 API 函数和其前后序列包含的 API 函数作为正样本,通过负采样技术在词汇表中抽取负样本。每个正样本经权重矩阵初始化后在隐藏层进行求和运算,得到词向量;再与输出权重矩阵进行相乘得到中心 API 前后序列的输出向量;经激活后得到该向量的概率分布,最大概率索引的 API 为 FastText 预测的 API 函数。

由此,最小化输出向量的目标函数由式(1)转变为

$$\frac{1}{k}\sum_{t=1}^{k}\log P(w_t \mid \boldsymbol{A}_{\text{Context}}(w_t)) \tag{3}$$

式中:$\boldsymbol{A}_{\text{Context}}$为正本集合;$P(w_t \mid \boldsymbol{A}_{\text{Context}}(w_t))$表示 w_t 为正样本的概率。

优化后正样本的似然函数为

$$g(w_t) = \prod \sigma(\boldsymbol{X}_w^{\mathrm{T}}\theta^a)^{L^w(a)}(1-\sigma(\boldsymbol{X}_w^{\mathrm{T}}\theta^a))^{(1-L^w(a))}$$
$$a \in \boldsymbol{A}_{\text{Context}}(w_t) \cup \boldsymbol{B}_{\text{NEG}}(w_t) \tag{4}$$

式中:$\boldsymbol{A}_{\text{Context}}$和$\boldsymbol{B}_{\text{NEG}}$分别表示正负样本集合;$\boldsymbol{X}_w$ 为正样本中各 API 函数的向量和;θ^a 为隐藏层与输出层间待优化的映射参数;$L^w(a)$用于判别分类情况,模型判定为正样本时为 1,否则为 0。

本文模型所用损失函数可进一步表示为

$$Loss = \sum (L^w(a) \times \log(\sigma(\boldsymbol{X}_w^{\mathrm{T}}\theta^a)) + (1-L^w(a)) \times \log(1-\sigma(\boldsymbol{X}_w^{\mathrm{T}}\theta^a))) \tag{5}$$

通过随机梯度上升法迭代更新 FastText 训练词向量时所需参数组为 $\boldsymbol{X}_w$ 和 θ^a,*Loss* 函数用于计算参数组的损失函数和梯度,$\boldsymbol{X}_w$ 是正样本中心 API 函数所对应的前后窗口中所有向量的加和。在迭代过程中,θ^a 的梯度加和会影响前后序列中 API 函数对应的向量,每个 API 函数对应的词向量也会根据 $\boldsymbol{X}_w$ 和 θ^a 进行更新。

在恶意代码家族分类的应用中,本文所提出的 FastText + CNN + LSTM 模型相较于传统 word2vec 模型具有以下优点:

1)由于引入了 N-gram 特征,使模型对低频 API 函数的映射加强了局部表征能力,提升了对低频和相似 API 函数的映射准确度,进而提升了模型性能;

2)对未收录在 token 词典中的 API 函数仍可从字符级的 N-gram 特征中构造词向量,因而可以对词典外 API 函数进行向量化,增加了模型的普适性。

2 FastText + CNN + LSTM 恶意代码家族分类模型的构建

本文在 FastText 框架下搭建模块化的深度学习分类模型 FastText + CNN + LSTM,该框架的恶意代码家族分类模型面向 Windows 系统文件,主要分为数据预处理模块、恶意代码特征预提取模块和深度学习模块。FastText 恶意代码家族分类模型示意如图 3 所示。

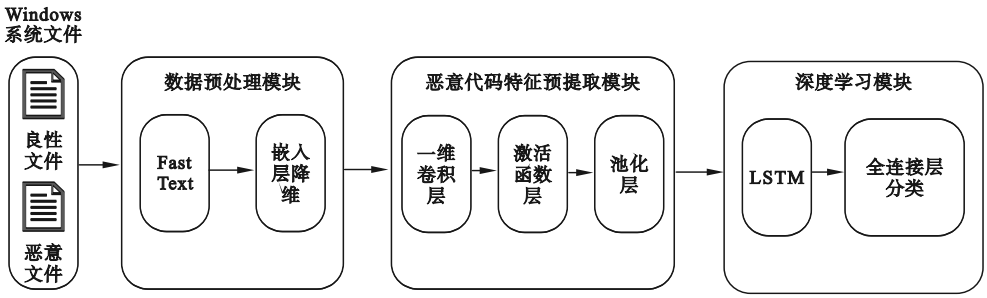

图 3 FastText 恶意代码家族分类模型示意图

Fig. 3 Schematic diagram of FastText malicious code family classification model

2.1 数据预处理模块

数据预处理模块的主要功能是获取样本文件并从文件 PE 头中提取该样本的 API 调用序列,最终在 FastText 中完成向量化。PE 头中提取特征保存为.json 格式,再转存到 Windows 系统的.csv 文件中;根据预先标注信息为数据添加标签;使用 NLP 工具进行分词、生成 token 词典等;使用 FastText 生成词向量。

传统的 word2vec 会把 token 词典中的每个单词视为一个基本单位,然后针对每一个基本单位训练该单位所对应的唯一多维向量,word2vec 忽略了单词内部形态特征的联系。例如在 Windows 系统创建网络链接和重定向到网络中常调用的 API 函数“WNetAddConnection2A”和“WNetAddConnection2W”,两个 API 函数功能完全一致,字符组成(内部形态)极为相近,但 word2vec 框架分别对其生成对应的向量,单词内部形态信息因为分别进行的向量化而丢失,导致对包含两个 API 函数的样本学习时产生偏差。

通过计算不同框架对相似 API 函数映射向量的距离和相似度,可以清晰对比两种模型处理相似 API 函数的性能区别。表 1 列出了 $n=7$ 时 FastText 框架对上述两个 API 函数取子词顺序,并对比了 word2vec 框架和 FastText 框架对两个 API 函数的映射词向量、向量间欧氏距离和向量间相似度。表中“<”和“>”分别代表前缀和后缀,是边界符号。

表 1　word2vec 词向量与 FastText 词向量对比表

Table 1　Comparison of word2vec word vector and FastText word vector

	WNetAddConnection2A	WNetAddConnection2W
FastText 框架 $n=7$ 时子词信息	“ < WNetAd ”、“ WNetAdd ”、“ NetAddC ”、“ etAddCo ”、“ tAddCon ”、“ AddConn ”、“ddConne”、“dConnec”、“Connect”、“onnecti ”、“ nnectio ”、“ nection ”、“ ection2 ”、“ction2A”、“tion2A > ”	“ < WNetAd ”、“ WNetAdd ”、“ NetAddC ”、“ etAddCo ”、“ tAddCon ”、“ AddConn ”、“ddConne”、“dConnec”、“Connect”、“onnecti ”、“ nnectio ”、“ nection ”、“ ection2 ”、“ction2W”、“tion2W > ”
生成词向量(word2vec)	[-0.038 612 89　0.019 049 0　-0.072 768　-0.022 517　0.023 756　0.042 353　…　-0.037 267 6　0.038 930　-0.004 051　-0.035 299 9　-0.005 116　-0.013 46]	[-0.049 195　0.0175 256　-0.094 0777　-0.036 531　0.023 710　0.017 58　…　-0.025 967 25　0.043 913 9　-0.016 750 6　-0.072 114　-0.017 879 3　-0.032 493]
生成词向量(FastText)	[0.320 040　-0.336 752 3　0.018 913 8　-0.342 861 8　0.170 803 6　-0.085 92　…　0.473 567 7　0.332 389 98　-0.281 093 1　-0.328 270 1　0.143 427　0.582 855]	[0.311 112 1　-0.342 722 1　0.028 833 3　-0.351 757　0.174 356 13　-0.088 20　…　0.483 091 65　0.324 358 1　-0.279 263　-0.338 78　0.139 166　0.587 299]
欧氏距离(word2vec)	0.128 144 13	
欧氏距离(FastText)	0.044 390 79	
相似度(word2vec)	0.935 706 14	
相似度(FastText)	0.999 995 65	

通过表 1 中拆解两个 API 函数所对应的子词信息可以看出,二者只有最后两个子词不同,本文模型先将 API 函数对应的子词映射为向量,再与词典中的 embedding 信息所映射的向量相加,映射成该 API 函数在 FastText 框架下的向量,送入后续模型进行训练。因此,字符组成上相近的 API 函数对应的词向量也较为相近,FastText 生成的两个词向量相似度高于 word2vec。通过表 1 中计算两者欧氏距离可发现,FastText 生成的相似 API 词向量具有更小的多维空间距离,相较于 word2vec 减小了约 60%,相似度提升约 6%,说明 FastText 可有效处理相似 API 函数之间的映射问题。

恶意代码程序的 API 序列包含操作较多,若将其转换为 one-hot 编码,稀疏性通常较大,不仅会导致算力浪费,还会影响模型的性能。因此,数据集向量化后需经过词嵌入层进行初步降维。嵌入层的核心思想是将高维的稀疏向量映射到低维的密集向量。在恶意代码 API 序列生成的字典中,经过 FastText 框架预处理后,共有 1 860 个单词(API 函数),嵌入层的输出维度与 FastText 相同,输出维度为 50,该层接受的序列最大长度为 200。

2.2　恶意代码特征预提取模块

词嵌入层降低了原始单词向量的维数,但向量维数仍然较高,LSTM 无法准确提取向量中的重要特征[21],导致算力的浪费和模型性能的低下。因此,数据在深度学习模块之前需要在特征预提取模块中处理,以提取 API 向量的初步特征,有助于提升训练速度和特征提取的精确度。该模块采用一维卷积层[22],其卷积核大小为 3 × 3,步长为 1,激活函数为 ReLU,输出矩阵维度为 200 × 128。

卷积层的参数误差引起的估计平均值偏差,是 API 词向量卷积运算的主要误差。采用最大池化层可以有效地避免该误差[23],并保留更多纹理信息用于后续学习。恶意代码的 API 序列中的非异常操作并不罕见,在良性代码中也很常见,而该特性会影响模型性能,在恶意代码分类领域加强背景纹理信息对于准确识别具有重要作用。在进一步提取最大池化层的特征后,网络所需的参数数量和算力将大大减少,也有利于防止网络的过拟合。卷积层采用下采样卷积,池化层的池化窗口为 2 × 2。

3　实验

3.1　实验环境

本文所实现的静态恶意代码家族分类针对 Windows 系统下的样本程序,神经网络模型采用 Keras 搭建。实验采用的硬件环境为因特尔酷睿 CPU(i7-10875H)、16 GB 内存。软件环境为

Windows10 系统、Python3.8、Tensorflow2.5.0 和 Keras 2.5.0。

3.2 数据集与预处理

本文使用 Ember 数据集,可以用于以静态方式训练恶意代码家族分类的机器学习模型,是恶意代码家族分类领域中常见的数据集。数据集包含 Ramnit、Lethic、Sality、Emotet 和 Ursnif 五个恶意代码家族及其变种,以及 2018 年及之前扫描的超过 100 万个 PE 文件的集合。PE 文件的功能被拆解并保存为.json 格式,包括 90 万个训练样本和 20 万个测试样本。数据集中的样本分布如表 2 所示。

表 2 数据集样本分布

Table 2 Dataset sample distribution

恶意代码家族	样本数量	恶意代码种类
Ramnit	186 000	Worm
Lethic	210 000	Trojan
Sality	190 000	Worm
Emotet	209 000	Trojan
Ursnif	300 000	Trojan

首先,读取包含 API 函数和标签的数据集,使用 jieba 分词工具提取 API 函数并将其转换为独立的单词,为后续提取字符集的 N-gram 特征提供基础。使用 Tokenizer 工具将分词后的文本转换为词典,本文生成的 token 词典如表 3 所示。

表 3 token 词典

Table 3 Token dictionary

序号	API 函数
0	GetDriveTypeA
1	ExitProcess
2	GetModuleHandleA
……	……
1860	FreeCredentialsHandle

在 FastText 框架下对输入样本进行向量化,本文生成词向量如表 4 所示。

FastText 生成的词向量矩阵如表 5 所示。

3.3 实验结果与分析

在恶意代码家族分类的研究中,模型对不同恶意代码家族的分类精确率不同,主要是因为数据集本身的不均衡性和不同家族的恶意代码为了达成主要目的所需 API 函数不同。

表 4 词向量

Table 4 Word vector

API 函数	词向量				
GetProcAddress	[-0.362 744 7	0.005 201 69	…	-0.060 137 19	1.245 243 8]
ExitProcess	[0.544 677 4	0.622 873 96	…	-1.125 845	0.303 592 98]
GetModuleHandleA	[-0.365 172 1	-0.736 485 4	…	-1.180 412 5	1.916 975 9]
TerminateProcess	[0.662 418 9	1.451 568 4	…	-1.037 370 2	-0.609 018 2]
LoadLibraryA	[0.854 843 5	-0.032 934 05	…	-0.493 055	0.916 790 96]
UnhandledExceptionFilter	[1.051 844	2.042 534 6	…	-1.131 622 4	-0.852 192 64]
GetLastError	[-0.345 186 1	-0.046 603 42	…	0.116 606 74	0.559 783 1]
…	…	…	…	…	…

表 5 FastText 词向量矩阵

Table 5 FastText word vector matrix

词向量矩阵				
[-0.244 876 47	-0.955 066 8	……	-3.857 153 89	1.386 756 3]
[-0.049 142	-0.384 862 6	……	-1.174 036 03	0.551 210 46]
-0.222 397 58	0.167 140 95	……	0.107 807 74	0.111 501 08]
……	……	……	……	……
[0.676 408 47	0.642 063 2	……	-0.213 219 12	-0.469 662 67]]

根据实验结果,首先研究 FastText + CNN + LSTM 模型提取的 N-gram 特征中 n 值对不同恶意代码家族识别精确度的影响,然后对比 word2vec + CNN + LSTM 模型和 FastText + CNN + LSTM 模型性能,最后对本文模型进行十折交叉验证。

FastText + CNN + LSTM 模型 N-gram 取词时,n 值在[1,10]区间内模型对数据集中五个恶意代码家族的分类精确率影响如图 4 和图 5 所示。

由图 4 和图 5 可以看出,本文模型对于 Lethic 家族的识别精确率最高,不同的 n 值对该家族影响较小,精确率始终为 98% 以上。该模型对 Emotet 家族和 Ursnif 家族的分类性能受 n 值影响较

大,在 n 取值为 2 和 4 时精确率最高,分别为 95% 和 92%。该模型在 Ramnit 家族和 Sality 家族上的识别精确率略低于其他家族,两个家族的分类精确率受 n 取值的影响较大,在 n 取 7 时对两个家族的识别精确率同时达到最大,分别为 85% 和 81%。综合考虑模型总体准确率和对每个恶意代码家族的识别精确率,设置 $n=6$,此时模型综合性能最符合实际需要。

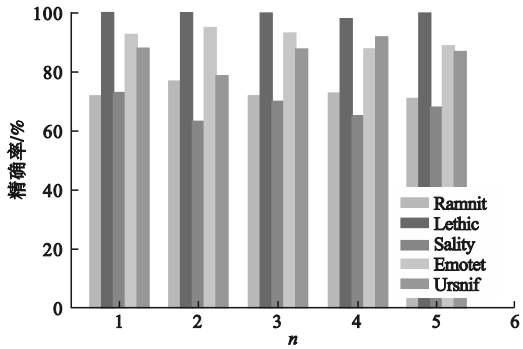

图 4　n 在[1,5]区间取不同值对模型的影响

Fig. 4　The influence of taking different values of n in the [1,5] interval on the model

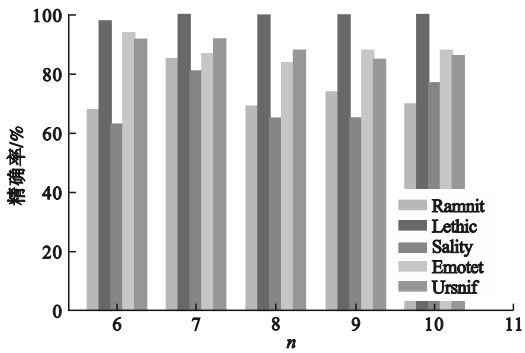

图 5　n 在[6,10]区间取不同值对模型的影响

Fig. 5　The influence of taking different values of n in the [6,10] interval on the model

本文为对比所选框架与传统 word2vec 框架的性能,汇总了几种模型的准确率和损失函数,如表 6 所示。

表 6　模型性能对比表

Table 6　Model performance comparison

模型	训练集准确率/%	训练集损失函数	测试集准确率/%	测试集损失函数
embedding + CNN + LSTM	71.30	0.683 8	70.30	0.788 6
word2vec + CNN + LSTM	95.32	0.089 1	86.14	0.779 4
FastText + CNN + LSTM	99.75	0.004 6	88.12	0.716 9

由表 6 可以看出,FastText + CNN + LSTM 的准确率和损失函数两项数据在训练集和测试集上的表现均优于其余两种模型,其准确率相较 embedding + CNN + LSTM 方法提升约 25%,相较 word2vec + CNN + LSTM 提升约 4.4%,最终在训练集上的准确率为 99.75%。word2vec + CNN + LSTM 和 FastText + CNN + LSTM 准确率对比如图 6 所示。

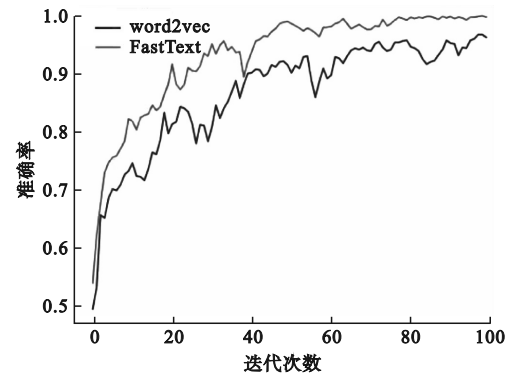

图 6　准确率对比图

Fig. 6　Accuracy comparison chart

由图 6 可见,word2vec 框架在训练过程中性能始终低于 FastText,主要是因为该框架只是对 API 序列的简单映射,造成对词典外和低频 API 函数映射失真。由于 FastText 引进字符级 N-gram 特征,训练相对平稳,模型的性能相较 word2vec 有所提升,最终准确率稳定在 99.75%。

图 7 为 word2vec + CNN + LSTM 模型和 FastText + CNN + LSTM 模型的损失函数对比图。

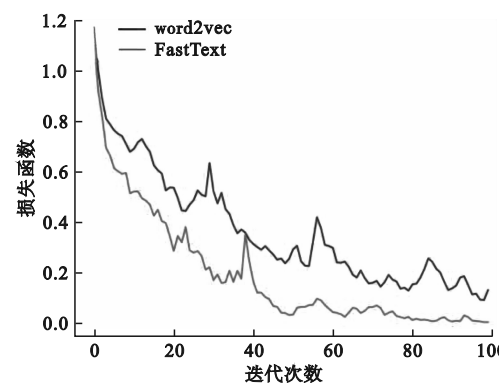

图 7　损失函数对比图

Fig. 7　Comparison diagram of loss function

由图 7 可见,在 40 次迭代前,二者的损失函数值均快速下降,但是在第 40 次迭代时两个模型性能的提升速度均有所放缓,FastText + CNN + LSTM 模型的性能均高于 word2vec + CNN + LSTM 模型,最终损失函数为 0.004 6。

图 8 为 FastText + CNN + LSTM 模型十折交叉验证结果。

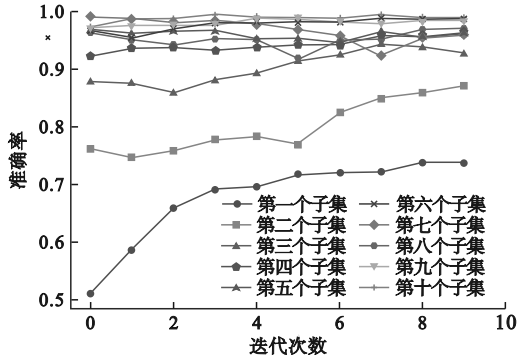

图 8 FastText + CNN + LSTM 模型十折交叉验证结果

Fig. 8 FastText + CNN + LSTM model 10-fold cross validation results

由图 8 可以看出,开始时模型性能较低,在第二个子集末期模型准确率可达 90%,在约四个子集后模型准确率可达 95% 以上,最终模型的准确率为 99%。从十折交叉验证的整体情况可知,该模型由于引入了字符级 N-gram 特征,模型收敛速度较快,且可以在性能较高的位置收敛,收敛准确率可满足恶意代码家族分类的实际需要。

4 结论

为解决恶意代码家族分类领域常见模型中对 API 调用序列向量化不准确、对低频及词典外 API 函数映射失真等问题而导致模型准确率无法进一步提升的问题,本文提出了一种基于 FastText 的恶意代码家族分类方法。通过提取样本代码所调用 API 序列的字符级 N-gram 特征,在负采样技术优化的 FastText 中生成词向量,使用词嵌入层进行初步降维,并使用 CNN + LSTM 模型进行特征提取和训练,最终在 Softmax 全连接层中输出样本所属类别的概率。通过对实验结果的分析,本文模型的性能高于传统 embedding 模型和 word2vec 模型,在与 word2vec 模型对比训练过程后证明 FastText 模型的训练过程平稳且收敛性更优,说明该模型在恶意代码家族分类领域具有较强的适应性,缓解了 word2vec 框架对低频和不均衡 API 函数映射失真的问题,进一步提升了恶意代码家族分类的准确率。

参考文献(References):

[1] 王博. 新形势下网络安全的挑战与应对策略[J]. 网络安全技术与应用,2022(10):95 - 96.
WANG B. Challenges and countermeasures of network security under the new situation[J]. Network Security Technology & Application,2022(10):95 - 96. (in Chinese)

[2] UROOJ U, ALISALEHAL-RIMYB, ZAINAL A, et al. Ransomware detection using the dynamic analysis and machine learning:a survey and research directions[J]. Applied Sciences,2021,12(1):172.

[3] 谢琦. 基于静态特征的恶意代码家族分类[D]. 长沙:国防科技大学,2021.

[4] CAVIGLIONE L, CHORAŚ M, CORONA I, et al. Tight arms race:overview of current malware threats and trends in their detection[J]. IEEE Access,2020,9:5371 - 5396.

[5] CUI Z H, DU L, WANG P H, et al. Malicious code detection based on CNNs and multi-objective algorithm[J]. Journal of Parallel and Distributed Computing,2019,129:50 - 58.

[6] 赵翠镕,张文杰,方勇,等. 基于语义 API 依赖图的恶意代码检测[J]. 四川大学学报(自然科学版),2020,57(3):488 - 494.
ZHAO C R, ZHANG W J, FANG Y, et al. Malware detection based on semantic API dependency graph[J]. Journal of Sichuan University (Natural Science Edition),2020,57(3):488 - 494. (in Chinese)

[7] 王博,蔡弘昊,苏旸. 基于 VGGNet 的恶意代码变种分类[J]. 计算机应用,2020,40(1):162 - 167.
WANG B, CAI H H, SU Y. Classification of malicious code variants based on VGGNet[J]. Journal of Computer Applications,2020,40(1):162 - 167. (in Chinese)

[8] 杨宇夏,孙皓月,高燚. 基于 N-grams 和灰度图特征融合的恶意代码检测方法[J]. 电脑知识与技术,2022,18(9):80 - 82.
YANG Y X, SUN H Y, GAO Y. Malicious code detection method based on N-grams and grayimage feature fusion[J]. Computer Knowledge and Technology,2022,18(9):80 - 82. (in Chinese)

[9] 王栋,杨珂,玄佳兴,等. 基于一维卷积神经网络的恶意代码家族多分类方法研究[J]. 计算机应用与软件,2021,38(12):332 - 336,340.
WANG D, YANG K, XUAN J X, et al. Research on multi classification method of malicious code family based on one dimension convolutional neural network[J]. Computer Applications and Software,2021,38(12):332 - 336,340. (in Chinese)

[10] EGITMEN A, BULUT I, AYGUN R C, et al. Combat mobile evasive malware via skip-gram-based malware detection[J]. Security and Communication Networks,2020,2020:1 - 10.

[11] 阴爱英,吴运兵,郑一江,等. 基于 FastText 模型的词向量表示改进算法[J]. 福州大学学报(自然科学版),2019,47(3):314 - 319.
YIN A Y, WU Y B, ZHENG Y J, et al. Base on FastText model to improve the word embedding of phrases and morphology[J]. Journal of Fuzhou University(Natural Science Edition),2019,47(3):314 - 319. (in Chinese)

[12] GHUKASYAN T, YESHILBASHYAN Y, AVETISYAN K. Subwords-only alternatives to FastText for morphologically rich languages[J]. Programming and Computer Software,2021,47(1):56 - 66.

[13] 李志明,孙艳,何宜昊,等. 融合类别特征扩展与 N-gram 子词过滤的 FastText 短文本分类[J]. 小型微型计算机系统,2022,43(8):1596 - 1601.

(下转第 90 页)