第 46 卷 第 11 期 计 算 机 工 程 2020 年 11 月
Vol. 46 No. 11 Computer Engineering November 2020

·网络空间安全· 文章编号：1000-3428(2020)11-0174-07 文献标志码：A 中图分类号：TP316

基于特征加权的深度学习 Android 恶意检测系统研究

葛文麒[a]，杨 清[a]，廖俊国[a]，何羽轩[b]

（湖南科技大学 a. 计算机科学与工程学院；b. 潇湘学院 计算机系，湖南 湘潭 411201）

摘 要： 当前 Android 系统恶意应用程序数量增长迅猛，然而传统检测系统无法对其进行快速有效检测，移动终端安全性面临严重威胁。提出一种将特征加权与双向长短期记忆（Bi-LSTM）神经网络深度学习算法相结合的恶意检测系统。采用静态分析方法从恶意与良性应用程序中提取不同类型行为特征，利用特征加权方法消除噪声与不相关因素后构建特征向量，使用 Bi-LSTM 深度学习算法优化行为特征参数，并设计恶意与良性应用程序分类模型，建立特征加权与深度学习算法相结合的恶意应用程序检测系统。实验结果表明，与支持向量机、RNN 等传统检测系统相比，该系统对恶意应用程序具有较高的检测精度与准确率。

关键词： Android 系统；恶意应用；特征加权；深度学习；双向长短期记忆神经网络

开放科学（资源服务）标志码（OSID）：

中文引用格式： 葛文麒，杨清，廖俊国，等. 基于特征加权的深度学习 Android 恶意检测系统研究[J]. 计算机工程，2020，46(11)：174-180.

英文引用格式： GE Wenqi，YANG Qing，LIAO Junguo，et al. Research on Android malware detection system using deep learning based on feature weighting[J]. Computer Engineering，2020，46(11)：174-180.

Research on Android Malware Detection System Using Deep Learning Based on Feature Weighting

GE Wenqi[a]，YANG Qing[a]，LIAO Junguo[a]，HE Yuxuan[b]

（a. School of Computer Science and Engineering；b. Department of Computer，Xiaoxiang College，Hunan University of Science and Technology，Xiangtan，Hunan 411201，China）

【Abstract】 At present the rapidly growing malicious applications in Android systems have imposed significant threats to the security of mobile terminals，but the traditional detection systems fail to detect them quickly and effectively. To address the problem，this paper proposes a malware detection system which combines feature weighting with the deep learning algorithm using Bidirectional Long Short-Term Memory（Bi-LSTM）neural network. The static analysis method is used to extract different types of behavior features from malicious and normal applications. The feature weighting method is used to eliminate noise and irrelevant factors to construct feature vectors. The Bi-LSTM-based deep learning algorithm is used to optimize the behavior feature parameters. Then a classification model for malicious and normal applications is designed，and on this basis a detection system for malicious applications combining feature weighting and the deep learning algorithm is constructed. Experimental results show that compared with traditional detection systems such as Support Vector Machine（SVM）and RNN，the proposed system has higher precision and accuracy in malicious application detection.

【Key words】 Android system；malware application；feature weighting；deep learning；Bidirectional Long Short-Term Memory（Bi-LSTM）neural network

DOI： 10.19678/j.issn.1000-3428.0056277

0 概述

移动终端因其便利性在人们生活中应用逐渐广泛，其安全问题也成为学者们关注的热点。从 2008 年 9 月第一部 Android 移动智能手机发布至今，移动终端的发展日趋迅猛，目前全球 Android 移动设备已超过 25 亿台[1]。由于移动终端是执行各类应用程序的开放平台，因此成为众多恶意攻击者的首

基金项目： 国家自然科学基金（61772194）。

作者简介： 葛文麒（1992—），男，硕士研究生，主研方向为移动网络安全；杨 清、廖俊国，教授、博士；何羽轩，本科生。

收稿日期： 2019-10-14 **修回日期：** 2019-12-10 **E-mail：** 1439233705@qq.com

选目标。自 2013 年以来,恶意应用程序在移动终端的增长速度远高于 PC 端[2]。

2018 年 IDC 报告显示,Android 系统以85.1%的市场份额位居操作系统榜首,其在操作系统市场已占据主导地位[3]。与其他移动操作系统相比,Android 系统具有开源性,其允许用户从第三方应用商店下载应用程序。然而有许多应用商店缺乏检测机制来识别恶意应用程序,导致此类应用程序呈增长趋势,对 Android 移动设备的安全性造成严重威胁。

目前研究人员主要采用基于数据跟踪的动态分析方法与基于反编译文件的静态分析方法对应用程序进行检测。动态分析方法需要在沙箱中运行应用程序,监视代码所有特征行为以及网络流量捕捉、文件加载等动态特性,该过程会消耗大量资源和时间,无法对应用程序进行快速检测。而静态分析方法是对应用程序安装包进行安全检测,无需运行程序。该方法通过逆向工程技术获得应用程序行为特征并从整体对应用程序进行分析,可快速高效识别恶意应用程序,但是不能运行应用程序做动态检测。在当前恶意应用程序数量快速增长的趋势下,与动态分析方法相比,采用静态分析方法对应用程序进行大规模分析更便捷,因此通常采用静态分析方法提取海量应用程序的行为特征,但是所提取特征中较多噪声与不相关行为特征会影响应用程序的检测性能[4-6]。

由于深度学习方法在图像与语音识别等领域具有优异性能,为消除上述行为特征中的噪声与不相关因素,本文采用基于深度学习的改进特征加权方法,分别从恶意与良性应用程序中提取行为特征,分析该行为特征并消除其中噪声与不相关的行为特征来构建特征向量,同时建立双向长短期记忆(Bidirectional Long Short-Term Memory, Bi-LSTM)神经网络模型对所提取特征信息进行自主学习,对参数进行优化分析以达到最佳检测性能,并对比了不同属性特征的检测效果。

1　相关工作

以下对用于检测 Android 系统中恶意应用程序的基于数据跟踪为主的动态分析方法、基于反编译文件分析为主的静态分析方法以及基于深度学习的检测方法的相关工作进行介绍。

1.1　静态分析方法

文献[7]针对目前 Android 系统防御机制不足的问题提出轻量级检测方法,但该方法利用应用程序的特征信息只能进行简单检测。文献[8]受生物学 DNA 序列比对思想启发,采用文本比较算法和套袋法相结合来评估系统所调用特征序列的相似性,对已知的恶意应用程序具有良好检测效果。文献[9]设计出 Android 多标准应用可信度评估器 MAETROID,仅利用程序元数据而无需分析代码,大幅降低检测复杂性,同时采用层次分析法对应用程序进行多标准决策组合分析并为用户提供安装建议。然而该检测方法仅针对权限特征来检测 Android 系统中恶意应用程序,无法全面反映行为信息。文献[10]采用静态分析技术从应用程序中提取检测信息,并利用多数表决法将多个分类器以投票的方式进行应用程序检测。该方法在游戏类应用程序检测中取得较好效果,但对其他类别应用程序检测效果很差,无法对未知应用程序进行全面检测。文献[11]提出贝叶斯分类检测模型,通过静态分析方法检测未知的 Android 恶意应用程序,该模型选取特征类型单一,无法进行全面有效检测。文献[12]开发出 Android 权限控制和推荐系统 RecDroid,从用户中广泛收集应用程序权限授予决策,同时采用贝叶斯学习模型评估用户专业水平并收集其权限控制决策,从而为其他用户提供依据。

1.2　动态分析方法

文献[13]在文献[12]的基础上,采用动态分析方法并将支持向量机(Support Vector Machine, SVM)与 K 最近邻(K-Nearest Neighbor, KNN)机器学习算法相结合构成应用程序评估模型,提升恶意应用程序检测性能,然而该模型会增加额外开销并占用大量资源。文献[14]提出 M0Droid 恶意应用程序检测模型,通过 Spearman 等级相关系数对恶意与良性应用程序中系统调用的行为信息进行相似性判别,但是若行为信息相近或恶意应用程序为未知,则不能进行有效检测。文献[15]构建基于应用程序行为特征的恶意应用程序动态检测模型 Crowdroid,通过众包搜寻应用程序在用户移动终端的运行状况,并在 Android 应用程序的文件系统、内核系统调用与事件检测模块捕获恶意应用程序的动态行为,该模型对使用频率较高的应用程序检测较准确,但无法判别使用频率较低的应用程序。文献[16]基于文献[17]的思想将隐私数据作为污染源,用动态污点分析技术监控移动终端的敏感信息并跟踪多个敏感数据源,以区分相同版本的良性与恶意应用,然而该方法在检测过程中会占用大量资源,且不能进行准确安全评估,无法快速有效地检测出恶意应用程序。

1.3　深度学习检测方法

传统恶意应用程序检测方法均为浅层结构,无法对恶意应用程序进行全面有效检测,由于深度神经网络(Deep Neural Networks, DNN)在图像识别和人工智能领域具有良好的检测性能,因此研究人员将深度学习方法应用于恶意应用程序检测。文献[18]提出 DeepClassifyDroid 检测系统,通过静态分析提取特征集,采用词嵌入技术将所提取特征输入卷积神经网络(Convolutional Neural

Networks,CNN)对应用程序进行分类检测,然而该系统仅选取静态特征建立特征集作为特征向量,未分析特征且增加检测开销。文献[19]采用长短期记忆(Long Short-Term Memory,LSTM)神经网络对恶意与良性应用程序的系统调用序列进行训练,通过计算其相似度评分对应用程序进行评估。文献[20]提出一种 Android 检测系统,使用深度卷积神经网络(Deep Convolutional Neural Networks,DCNN)检测原始操作码序列,减少人工提取特征造成的误差。

上述基于深度学习的检测方法仅从应用程序提取特征信息,未对特征信息进行充分分析,降低了应用程序检测准确性。此外,由于动态分析方法会增加训练模型时间、占用大量计算资源等额外开销,无法对未知应用程序进行快速检测,因此本文采用静态分析方法构建基于特征加权的深度学习 Android 恶意检测系统,以消除冗余与不相关特征,增强恶意应用程序区分能力并减少额外开销,同时全面检测应用程序,自动挖掘其深层特征并分析特征之间的信息,以对 Android 应用程序进行快速准确检测。

2 系统设计

本文系统主要包括特征提取、特征选择、特征嵌入和分类检测 4 个部分,系统结构如图 1 所示。其中:特征提取是对应用程序进行广泛静态分析,通过逆向工程技术从中提取 6 种不同类型静态行为特征作为特征集,以防止行为特征单一导致检测准确率较低;特征选择是在提取行为特征过程中,为消除行为特征冗余与不相关性,利用改进特征加权的方法,将 TF-IDF 算法与信息增益相结合,按照权重选择行为特征构成特征集;特征嵌入是将不同维度的特征按权重高低选择特征并映射到联合向量空间;分类检测是在传统机器学习模型缺乏有效行为特征学习能力且无法有效分析大量行为特征信息的情况下,将转换得到的多维度特征向量作为输入,通过双向长短期记忆神经网络利用隐藏层节点学习到本质行为特征后,对未知应用程序进行分类,最终实现 Android 恶意应用程序检测。

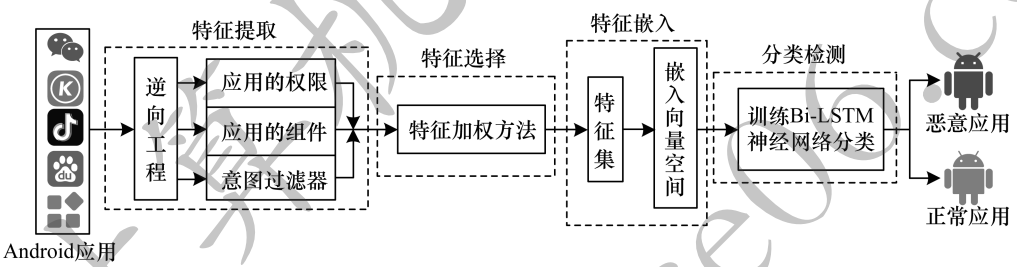

图 1 本文系统结构

Fig. 1 The proposed system structure

2.1 特征向量构建

2.1.1 特征选择

为有效检测 Android 系统恶意应用程序,本文从 Android 系统恶意与良性应用程序中提取全面而独特的行为特征,使用 APKTOOL 工具对 Android 系统应用程序的安装文件进行反编译生成 AndroidManifest.xml 文件,从中提取 Android 系统应用程序静态行为特征如下:

1)权限。权限是 Android 系统应用程序重要的安全机制之一。用户在安装 Android 系统应用程序时进行授权,使应用程序可访问不同类型的安全相关资源与数据。与正常应用程序相比,Android 系统恶意应用程序会请求申请更多权限以获得尽可能多资源和数据,因此,部分恶意应用程序可通过权限申请进行识别。

2)应用组件。Android 系统应用程序组件包括活动、数据通信、服务与广播。Android 系统应用程序可在系统清单文件中申请多个不同类型组件,其中恶意应用程序会在用户不知情的情况下通过服务组件在系统后台运行某些服务进程来执行恶意行为。

3)意图过滤器。Android 系统根据应用程序所配置意图过滤器的动作和类别进行比配,寻找响应意图的组件或服务,该机制为 Android 系统恶意应用程序监控用户移动终端提供便利,恶意应用程序在移动终端重启后会直接执行恶意行为。

2.1.2 选择算法

在提取的行为特征中,由于存在大量冗余与不相关特征导致恶意应用程序检测准确率降低,因此需要对行为特征进行分析。本文采用特征加权方法,结合 TF-IDF 算法与信息增益,考虑恶意与良性应用程序中每个行为特征的重要程度,并考虑上述行为特征与恶意或良性应用程序的关联程度。通过减少均匀存在于恶意与良性应用程序中调用行为特征的权重,来增强特定存在于恶意与良性应用程序中的行为特征。

假设恶意与良性应用程序所提取行为特征 S_i 被调用的频率分别为 $T_{(S_i,M)}$ 与 $T_{(S_i,B)}$,表达式分别如下:

$$T_{(S_i,B)}=\frac{N(S_i,B)}{N(B)} \tag{1}$$

$$T_{(S_i,M)}=\frac{N(S_i,M)}{N(M)} \tag{2}$$

其中,$N(S_i,M)$与 $N(S_i,B)$分别表示行为特征 S_i 在恶意与良性应用程序集中被调用的次数,$i=1,2,\cdots,n$,$N(M)$和 $N(B)$分别表示行为特征 S_i 在恶意与良性应用程序集中被调用的总次数,n 表示收集的行为特征总数。

将应用程序总数 N 分别除以恶意与良性应用程序中系统调用的行为特征 S_i,所得结果取对数得到的 $D_{(S_i,M)}$与 $D_{(S_i,B)}$分别表示行为特征 S_i 在恶意与良性应用程序中重要程度,表达式分别如下:

$$D_{(S_i,M)}=\ln\frac{N}{n(S_i,M)} \tag{3}$$

$$D_{(S_i,B)}=\ln\frac{N}{n(S_i,B)} \tag{4}$$

其中,$n(S_i,M)$与 $n(S_i,B)$分别表示恶意与良性应用集中调用特征 S_i 的应用程序个数。

将 $T_{(S_i,M)}$与 $D_{(S_i,M)}$相乘可得到行为特征 S_i 在恶意应用程序中权重,同理求出其在良性应用程序中权重,并将该权重与恶意应用程序中权重相减求绝对值,所得 $W(S_i)$表示行为特征 S_i 对区分恶意与良性应用程序的重要性,计算公式如下:

$$W(S_i)=\left|T_{(S_i,M)}\times D_{(S_i,M)}-T_{(S_i,B)}\times D_{(S_i,B)}\right| \tag{5}$$

本文中信息增益代表每个行为特征在恶意与良性应用程序集中带来的信息数量,信息数量越多,该行为特征越重要,其熵值也越高。对行为特征 S_i 而言,其在应用程序分类中熵值会发生变化,熵值在变化前后的差值表示行为特征在应用程序检测中重要程度,表达式如下:

$$\mathrm{IG}(S_i)=\sum_{X=(M,B)}\left(\sum_{Y=(S_i,\overline{S_i})}\left(\frac{n(Y,X)}{N}\ln\frac{n(Y,X)}{n(Y,N)}\right)\right)-\sum_{X=(M,B)}\left(\frac{N(X)}{N}\ln\frac{N(X)}{N}\right) \tag{6}$$

其中,$n(Y,X)$表示恶意或良性应用程序数目,X 表示恶意或良性应用程序集,$\overline{S_i}$表示未被调用的行为特征。

综上所述,通过行为特征 S_i 在恶意与良性应用程序之间重要程度 $W(S_i)$以及行为特征 S_i 区分恶意与良性应用程序的关联程度 $\mathrm{IG}(S_i)$可得到行为特征 S_i 的权重 $\varphi(S_i)$,计算公式如下:

$$\varphi(S_i)=W(S_i)\times\mathrm{IG}(S_i) \tag{7}$$

通过上述特征加权方法可求得不同特征集中每个行为特征 S_i 的权重 $\varphi(S_i)$,用其表示行为特征 S_i 对应用程序检测影响程度,并由此选择行为特征。

2.1.3　特征向量

应用程序的恶意行为会反映在系统调用特征集的行为特征上,因此,在检测恶意应用程序时,可使用单一特征集或者组合特征集。在使用组合特征集时,需将不同维度特征集组合为统一形式。使用特征加权方法计算得到不同特征集中每个行为特征 S_i 对检测影响的权重 $\varphi(S_i)$,按照权重 $\varphi(S_i)$由大到小从不同特征集中选出特征形成新特征集 S,表达式如下:

$$S=S(\varphi(S_p))\cup S(\varphi(S_I))\cup S(\varphi(S_c)) \tag{8}$$

其中,S_p 表示权限集合,S_I 表示意图过滤器集合,S_c 表示活动、数据通信、服务、广播四大组件集合。

本文将特征集 S 定义为具有 $|S_i|$维数的布尔表达式,并将其嵌入向量空间 $\boldsymbol{X}$ 中得到统一形式。假设恶意应用程序 F 在特征集中使用某些特征,则该特征集是 1 的向量,其位置为 0。因此,将任意应用程序转换为向量空间 $\boldsymbol{X}$,表达式如下:

$$\boldsymbol{X}=\{S_1,S_2,\cdots,S_k\}$$
$$k\in|S_i|$$
$$S_k=\begin{cases}1,F\in S_k\\0,F\notin S_k\end{cases} \tag{9}$$

通过上述方式,可将不同的特性集嵌入到统一的联合向量空间,利用组合特征集进行范围更广泛的检测。

2.2　改进的 Bi-LSTM 算法

2.2.1　检测模型

Bi-LSTM 算法是应用较广泛的深度学习算法,其由多个长短期记忆网络组成,在图像识别、文本分析等领域具有良好的检测效果,因此,本文将 Bi-LSTM 设计为分类网络结构,对使用特征加权方法生成的特征向量进行分类,从而深层分析上下文行为特征之间的信息,同时解决梯度消失与爆炸问题,并通过 Bi-LSTM 分类模型检测恶意应用程序,该模型结构如图 2 所示。首先使用特征加权方法构造特征向量作为输入,分析向量输入参数与系统恶意应用检测准确率的关系,并通过不断调节向量大小实现对恶意应用程序全面检测,然后分析不同属性特征向量对系统恶意应用程序的检测性能并选出最佳特征向量。Bi-LSTM 分类模型的隐藏层部分由 Bi-LSTM 组成,由于 Adam 算法是利用迭代次数和延迟因子对梯度均值与梯度平方均值进行校正,能准确预测梯度变化,收敛速度很快,因此本文训练 Bi-LSTM 网络时使用 Adam 算法通过迭代更新权重对其进行优化,并不断选取和调试 Bi-LSTM 网络隐藏层单元数以达到最优检测效果。在隐藏层后构造全连接层,由于整个过程执行分类检测任务,因此将全连接层输出作为分类层的输入,通过 Sigmoid 分类器对应用程序进行检测分类,最终使用二进制交叉熵损失函数评估检测模型预测效果。

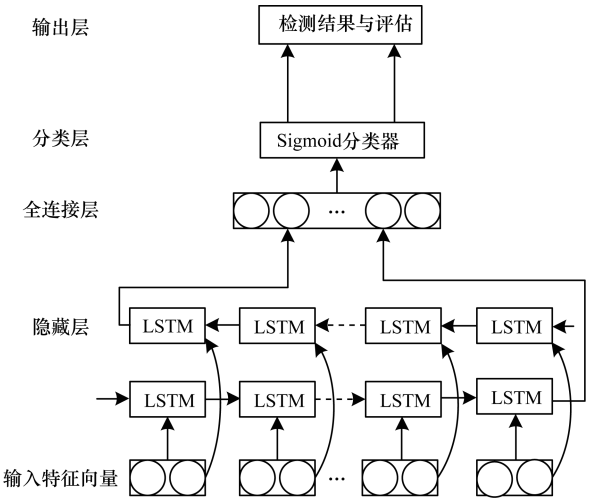

图 2 Bi-LSTM 检测模型结构

Fig. 2 Bi-LSTM detection model structure

2.2.2 Bi-LSTM 算法

LSTM 网络[21]由遗忘门、输入门、输出门和记忆单元构成,主要作用是对行为特征信息进行筛选、保存与更新,其结构如图 3 所示。为提升检测准确性,从前后两个方向分析所提取的信息,采用双向 LSTM 将上一个记忆单元状态(细胞状态)同时引入到输入门、遗忘门与更新计算中。

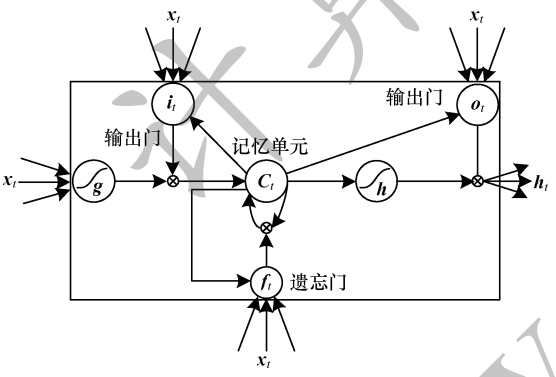

图 3 LSTM 网络结构

Fig. 3 LSTM network structure

在图 3 中,遗忘门对上一层细胞状态进行筛选,留下有用信息并遗忘无用信息,计算公式如下:

$$\boldsymbol{f}_t=\sigma(\boldsymbol{W}_f\times[\boldsymbol{h}_{t-1},\boldsymbol{x}_t]+\boldsymbol{b}_f) \tag{10}$$

其中,$\boldsymbol{W}_f$ 和 $\boldsymbol{b}_f$ 分别为遗忘门的权重和偏置,$\boldsymbol{h}_{t-1}$ 为上一层隐藏状,σ 为 Sigmoid 激活函数。遗忘门通过采用 Sigmoid 函数控制遗忘门,使其根据上一时刻输出 $\boldsymbol{h}_{t-1}$和当前输入 $\boldsymbol{x}_t$ 产生 $\boldsymbol{f}_t$ 值,并决定是否让上一时刻所得信息 $\boldsymbol{C}_{t-1}$通过。

利用输入门对信息进行判断,将重要信息送入细胞状态更新处以完成细胞状态更新,计算公式如下:

$$\boldsymbol{i}_t=\sigma(\boldsymbol{W}_i\times[\boldsymbol{h}_{t-1},\boldsymbol{x}_t]+\boldsymbol{b}_i) \tag{11}$$

$$\boldsymbol{C}_t=\boldsymbol{f}_t\times\boldsymbol{C}_{t-1}+\boldsymbol{i}_t\times\tanh(\boldsymbol{W}_c\times[\boldsymbol{h}_{t-1},\boldsymbol{x}_t]+\boldsymbol{b}_c) \tag{12}$$

其中,$\boldsymbol{W}_i$ 和 $\boldsymbol{b}_i$ 分别为输入门的权重和偏置,$\boldsymbol{W}_c$ 和 $\boldsymbol{b}_c$ 分别为细胞状态的权重和偏置,$\boldsymbol{i}_t$ 代表输入层,$\boldsymbol{C}_{t-1}$ 和 $\boldsymbol{C}_t$ 分别为原细胞状态和更新后细胞状态。

确定更新信息的过程由两阶段组成:1)利用 Sigmoid 函数确定需更新加入到细胞状态的信息;2)利用 tanh 激活函数将需更新的信息转换为可加入到细胞状态的候选向量$\boldsymbol{C}_{t-1}$,通过这两阶段生成新细胞状态 $\boldsymbol{C}_t$。

输出门包含当前输入、上一个隐状态、当前细胞状态等,主要是控制该层细胞状态输出,计算公式如下:

$$\boldsymbol{o}_t=\sigma(\boldsymbol{W}_o\times[\boldsymbol{h}_{t-1},\boldsymbol{x}_t]+\boldsymbol{b}_o) \tag{13}$$

$$\boldsymbol{h}_t=\boldsymbol{o}_t\times\tanh(\boldsymbol{C}_t) \tag{14}$$

其中,$\boldsymbol{W}_o$ 和 $\boldsymbol{b}_o$ 分别为输出门的权重和偏置。使用 Sigmoid 激活函数确定需输出的信息 $\boldsymbol{o}_t$,然后用 tanh 激活函数对细胞状态值进行缩放作为信息筛选条件,并对信息 $\boldsymbol{o}_t$ 筛选后输出所需信息 $\boldsymbol{h}_t$。

3 实验与结果分析

3.1 数据集

为检测本文模型的有效性,从 VirusShare 网站和 Android 官方商店下载应用程序,删除无法使用和重复的应用程序后分别得到 4 000 个和 3 500 个应用程序,将其依次标记为恶意与良性应用程序,上述两部分应用程序组成本文实验的数据集。

3.2 评价指标

本文实验所用处理器为 Intel® Core™ i3-2130,CPU 为 3.40 GHz,内存为 16 GB(RAM),采用 Windows 7 操作系统。以精确率(P)、召回率(R)、F1 值(F)和准确率(A)作为本文系统检测效果的评价指标,计算公式如下:

$$P=\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \tag{15}$$

$$R=\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \tag{16}$$

$$F=2\times\frac{P\times R}{P+R} \tag{17}$$

$$A=\frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{FP}+\mathrm{TN}+\mathrm{FN}} \tag{18}$$

其中,TP 为正确预测的恶意应用程序数量,TN 为正确预测的良性应用程序数量,FP 为错误预测的恶意应用程序数量,FN 为错误预测的良性应用程序数量。

3.3 参数设置

在 Android 系统应用程序数据集中随机选取 2 500 个良性应用程序与 3 000 个恶意应用程序,从

不同方面分析 Bi-LSTM 神经网络的结构性能。选择隐藏层单元数时,如果隐藏层单元数太少,则无法对数据进行训练或者造成检测性能很差无法准确检测应用程序;如果选择隐藏层单元数过多,则训练数据时容易陷入局部极小值而无法得到最优性能,并在训练时出现过拟合问题。在选取同一特征向量长度作为输入时,由于隐藏层单元数为 300 时系统检测性能较好,因此从隐藏层单元数为 300 时开始测试。图 4 为系统检测准确率与召回率随隐藏层单元数的变化曲线,可以看出在不同隐藏层单元数下系统准确率和召回率均分别达到 0.93 和 0.90 以上,准确率和召回率较高。当隐藏层单元数为 900 时,系统准确率和召回率分别为 0.947 5 和 0.943 2,相较其他隐藏层单元数下更高,因此本文选择 900 作为隐藏层单元数。

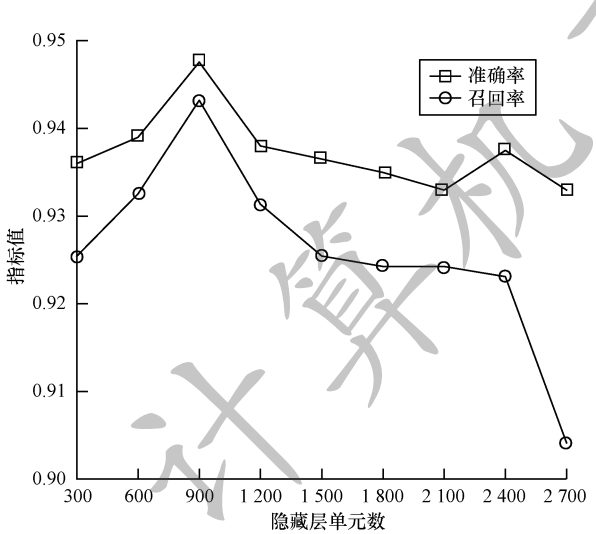

图 4　系统准确率与召回率随隐藏层单元数的变化曲线

Fig. 4　Variation curve of accuracy rate and recall rate of system with the number of hidden layer units

考虑到所输入特征向量长度对系统检测性能的作用,本文研究了不同特征向量长度对 Android 系统应用程序检测准确性的影响。由于输入特征集过少不能覆盖所有恶意和良性行为对检测效率的影响,而输入特征向量过长行为特征会受到额外的噪声干扰,由于特征向量长度为 1 000 时系统检测效果较好,因此将所输入特征向量长度初始值设置为 1 000。利用特征向量长度对 Android 系统恶意应用程序检测性能的影响,选出分类任务最优的特征向量。图 5 为系统检测准确率与召回率随特征向量长度的变化曲线,可以看出不同特征向量长度下系统准确率和召回率均达到 0.91 以上,准确率和召回率较高。当特征向量长度为 8 000 时,系统准确率和召回率分别为 0.953 1 和 0.958 6,相较其他特征向量长度下更高,因此本文选择 8 000 作为特征向量长度。

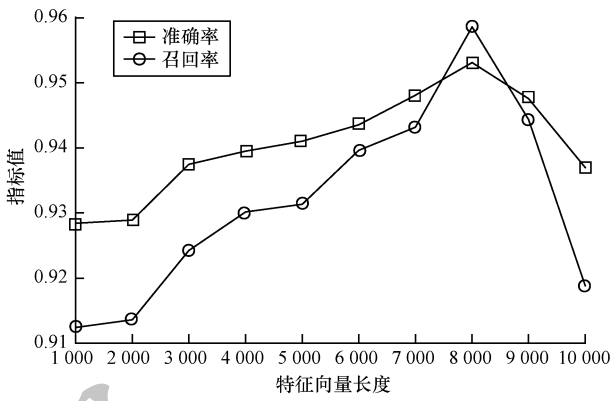

图 5　系统准确率与召回率随特征向量长度的变化曲线

Fig. 5　Variation curve of accuracy rate and recall rate of system with the length of eigenvector

3.4　特征集选择

为评估本文系统在不同特征集中对应用程序的检测效果,以使用逆向工程技术提取的 6 类不同静态行为特征作为特征集,将所有特征集放入联合向量空间形成组合特征集,并通过式(9)构建特征向量以更全面地分析。表 1 为不同特征集得到的评估结果,可见组合特征集对恶意应用程序检测效果最好,指标值高于其他单一类型特征集,这是因为单一类型特征集对应用程序检测提供信息较少,无法实现准确检测,因此本文选择全部特征构成的特征集检测应用程序。

表 1　不同特征集所得评估结果

Table 1　The evaluation results of different feature sets

特征集	*P*	*R*	*F*	*A*
权限	0.908 1	0.893 9	0.901 0	0.918 5
意图过滤器	0.902 6	0.773 6	0.833 1	0.871 1
活动	0.903 9	0.926 5	0.915 1	0.926 8
数据通信	0.869 6	0.899 4	0.884 3	0.856 5
服务	0.910 5	0.880 6	0.895 3	0.892 7
广播	0.877 1	0.940 3	0.907 6	0.904 4
组合	0.933 2	0.958 6	0.945 7	0.953 1

3.5　实验结果

为验证本文提出的基于特征加权深度学习算法的 Android 恶意检测系统检测效果,将 SVM、Logistic、Decision Tree、Naive Bayes 等机器学习算法与 RNN、LSTM、Bi-LSTM 等深度学习算法的恶意应用程序检测性能进行对比,结果如表 2 所示。可以看出上述两类算法的检测准确率分别超过 0.87 和 0.94,均具有良好的检测效果。而本文系统采用的 Bi-LSTM 算法所得指标值较其他算法更高,有效提升了恶意应用程序检测性能。

表 2 不同算法所得系统检测性能指标结果

Table 2 Detection performance index results of different algorithms

算法	*P*	*R*	*F*	*A*
SVM 算法	0.908 9	0.776 5	0.837 5	0.874 8
Logistic 算法	0.910 4	0.953 2	0.931 3	0.945 3
Decision Tree 算法	0.881 1	0.922 2	0.901 2	0.915 9
Naive Bayes 算法	0.892 0	0.910 9	0.901 4	0.917 1
RNN 算法	0.928 5	0.937 3	0.932 9	0.942 5
LSTM 算法	0.928 1	0.954 8	0.941 3	0.948 9
Bi-LSTM 算法	0.933 2	0.958 6	0.945 7	0.953 1

4 结束语

本文提出一种结合特征加权方法和双向长短期记忆神经网络的 Android 恶意检测系统。采用静态分析技术从 Android 系统应用程序中提取静态行为特征,利用特征加权方法分析行为特征和恶意与良性应用程序的关联程度,以去除冗余与不相关行为特征,并使用长短期记忆神经网络算法对所提取特征信息进行学习、分类与参数优化。实验结果表明,该系统具有较高的检测准确率,对恶意应用具有较强的识别能力,可大范围检测 Android 系统恶意应用程序。下一步将根据功能划分应用程序类别,提取对应的行为特征,并在特征集中扩展加入网络流量、动态文件加载等动态特征,以全面有效地提高系统检测准确率。

参考文献

[1] Project Euphonia: helping everyone be better undestood [EB/OL]. [2019-09-01]. https://www.zgwdy.cn/21043133610.html.

[2] UNDERWOOD B, BIRDSALL J, KAY E. The use of a mobile app to motivate evidence-based oral hygiene behaviour [EB/OL]. [2019-09-01]. https://www.nature.com/articles/sj.bdj.2015.660.

[3] KE Dongxiang, PAN Limin, LUO Senlin, et al. Android malicious behavior recognition and classification method based on random forest algorithm [J]. Journal of Zhejiang University (Engineering Science Edition), 2019, 53(10): 2013-2023. (in Chinese)
柯懂湘, 潘丽敏, 罗森林, 等. 基于随机森林算法的 Android 恶意行为识别与分类方法[J]. 浙江大学学报(工学版), 2019, 53(10): 2013-2023.

[4] ZHOU Yajin, JIANG Xuxian. Dissecting Android malware: characterization and evolution[C]//Proceedings of 2012 IEEE Symposium on Security and Privacy. Washington D.C., USA: IEEE Press, 2012: 95-109.

[5] VINOD P, ZEMMARI A, CONTI M. A machine learning based approach to detect malicious Android apps using discriminant system calls[J]. Future Generation Computer Systems, 2018, 94(11): 333-350.

[6] FEIZOLLAH A, ANUAR N B, SALLEH R, et al. A review on feature selection in mobile malware detection [J]. Digital Investigation, 2015, 13(6): 22-37.

[7] SATO R, CHIBA D, GOTO S. Detecting Android malware by analyzing manifest files [EB/OL]. [2019-09-01]. https://www.researchgate.net/publication/272778915_Detecting_Android_Malware_by_Analyzing_Manifest_Files.

[8] VIDAL J M, MONGE M A S, VILLALBA L J G. A novel pattern recognition system for detecting Android malware by analyzing suspicious boot sequences [J]. Knowledge-Based Systems, 2018, 150(6): 198-217.

[9] DINI G, MARTINELLI F, MATTEUCCI I, et al. Risk analysis of Android applications: a user-centric solution[J]. Future Generation Computer Systems, 2018, 80(3): 505-518.

[10] WANG Wei, LI Yuanyuan, WANG Xing, et al. Detecting Android malicious apps and categorizing benign apps with ensemble of classifiers [J]. Future Generation Computer Systems, 2018, 78(1): 987-994.

[11] YERIMA S Y, MCWILLIAMS G, SEZER S. Analysis of Bayesian classification-based approaches for Android malware detection[J]. IET Information Security, 2014, 8(1): 25-36.

[12] RASHIDI B, FUNG C, VU T. Android fine-grained permission control system with real-time expert recommendations[J]. Pervasive and Mobile Computing, 2016, 10(32): 62-77.

[13] REHMAN Z U, KHAN S N, MUHAMMAD K, et al. Machine learning-assisted signature and heuristic-based detection of malwares in Android devices[J]. Computers and Electrical Engineering, 2017, 69(11): 828-841.

[14] DAMSHENAS M, DEHGHANTANHA A, CHOO K K R, et al. M0Droid: an Android behavioral-based malware detection model [J]. Journal of Information Privacy and Security, 2015, 11(3): 141-157.

[15] BURGUERA I, ZURUTUZA U, NADJM T S. Crowdroid: behavior-based malware detection system for Android[C]//Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices. New York, USA: ACM Press, 2011: 15-26.

[16] ENCK W, GILBERT P, HAN S, et al. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones [J]. ACM Transactions on Computer Systems, 2014, 32(2): 5-6.

[17] ENCK W, PETER G, CHUN B G, et al. TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones [J]. ACM Transactions on Computer Systems, 2010, 57(3): 393-407.

[18] ZHANG Yi, YANG Yuexiang, WANG Xiaolei. A novel Android malware detection approach based on convolutional neural network [C]//Proceedings of the 2nd International Conference on Cryptography, Security and Privacy. New York, USA: ACM Press, 2018: 144-149.

[19] XIAO X, ZHANG S F, MERCALDO F, et al. Android malware detection based on system call sequences and LSTM[J]. Multimedia Tools and Applications, 2019, 78(4): 3979-3999.

[20] MCLAUGHLIN N, MARTINEZ D R J, KANG B, et al. Deep Android malware detection [C]//Proceedings of the 7th ACM Conference on Data and Application Security and Privacy. New York, USA: ACM Press, 2017: 1735-1780.

[21] HOCHREITER S, SCHMIDHUBER J. Long short-term memory[J]. Neural Computation, 1997, 9(8): 1735-1780.

编辑 宋 圆