doi: 10.3969/j.issn.0490-6756.2019.03.011

基于文本嵌入特征表示的恶意软件家族分类

张 涛, 王俊峰
(四川大学计算机学院, 成都 610065)

摘 要: 自动化、高效率和细粒度是恶意软件检测与分类领域目前面临的主要挑战. 随着深度学习在图像处理、语音识别和自然语言处理等领域的成功应用,其在一定程度上缓解了传统分析方法在人力和时间成本上的巨大压力. 因此本文提出一种自动、高效且细粒度的恶意软件分析方法-mal2vec,其将每个恶意软件看成是一个具有丰富行为语义信息的文本,文本的内容由恶意软件动态执行时的 API 序列构成,采用经典的神经概率模型 Doc2Vec 对文本集进行训练学习. 实验结果表明,与 Rieck[1] 等人的分类效果相比,本文方法得到的效果有明显提升. 特别的,不同于其他深度学习的方法,本文方法能够抽取模型训练的中间结果进行显式表示,这种显式的中间结果表示具有可解释性,可以让我们从细粒度层面分析恶意软件家族的行为模式.

关键词: 恶意软件; 分类; 文本嵌入; Doc2Vec
中图分类号: TP309.7 **文献标识码**: A **文章编号**: 0490-6756(2019)03-0441-09

Malware family classification based on text embedding feature representation

ZHANG Tao, WANG Jun-Feng
(College of Computer Science, Sichuan University, Chengdu 610065, China)

Abstract: Automation, efficiency, and granularity are major challenges in the area of malware detection and classification. With the successful application of deep learning in the fields of image processing, speech recognition and natural language processing, it has alleviated the enormous pressure of traditional analysis methods on manpower and time cost to some extent. This paper describes mal2vec: an automatic, efficient and fine-grained malware analysis method, which treats each malware as a text with rich behavioral semantic information. The content of the text is composed of API sequences when malware is dynamically executed. We use the classical neural probability model Doc2Vec to train the text set. The experimental results show that the effect of this paper is significantly improved compared with the classification effect of Rieck et al. In particular, unlike other methods of deep learning, this method can extract the intermediate results of model training for explicit representation. This explicit intermediate result is interpretable and allows us to analyze the behavior patterns of the malware family from a fine-grained level.

Keywords: Malware; Classification; Text Embedding; Doc2Vec

收稿日期: 2018-08-09
基金项目: 国家重点研发计划项目(2018YFB0804503, 2016QY06X1205); 装备预研教育部联合基金(6141A02011607, 6141A02033304); 四川省重点研发计划项目(18ZDYF3867, 2017GZDZX0002)
作者简介: 张涛(1992-), 男, 硕士生, 研究方向为网络与信息安全. E-mail: 572480349@qq.com
通讯作者: 王俊峰. E-mail: wangjf@scu.edu.cn

1 引 言

随着互联网的高速发展,互联网安全问题也受到愈来愈多的重视. 大多数的互联网安全问题都是由恶意软件引起的,如病毒、蠕虫、后门、僵尸网络和特洛伊木马. 恶意软件开发人员制造恶意软件的门槛越来越低,这些恶意软件开发人员可以轻松地使用网络上的各种恶意软件核心源码和其他修改工具在短时间内生成大量恶意软件,给社会甚至国家带来了巨大威胁. 传统的恶意软件检测和分类手段采用单一或组合的静态、动态特征[2],如可打印字符串、反汇编操作码、系统调用以及网络数据包等,构建一个庞大的特征矩阵,然后通过特征选择、降维等步骤,最后输入到二元判别模型或多分类模型完成检测和分类. 然而,这种方法存在几个严重的缺陷:1) 提取特征凭人工经验,需要足够的专业领域知识,且不是所有的特征都方便用一种量化的数值来表示;2) 每个步骤分离,不利于自动化;3) 数据规模小,无法应对日益指数增长的大规模恶意软件. 针对以上几种缺陷,我们设计并实现了一套完整的自动化恶意软件分类方案,追踪并提取恶意软件运行时的 API 调用序列,构建恶意软件知识库,通过一个浅层的神经网络模型进行训练并将恶意软件量化表示,最终将量化的恶意软件输入到分类器中进行家族分类. 实验结果表明,该方法能够有效地学习到恶意软件行为语义的信息,该信息对于恶意软件分析的其他应用场景如分类、家族行为模式的分析都具有重要意义.

2 相关工作

随着机器学习在图像处理、语音识别和自然语言处理等领域的成功应用,机器学习成为近 10 年来恶意软件分析的重要手段. 研究人员和反病毒公司早已开始采用机器学习算法来解决这个问题. 在学术界,研究人员已经提出了使用各种各样的机器学习分类器来用于恶意软件分类[3-9],如逻辑回归[3]、神经网络[4]以及决策树[5]. Schultz[6]等人提取了三种不同的静态特征:动态链接库(DLL)和 API 函数、字符串、字节序列,然后使用三种算法分别处理三类特征,得到能区分良性软件和恶意软件的模型. 此后,Kolter[7]等改进了文献[6]的第三类特征,使用部分重叠的字节序列 n-grams 代替非重叠的字节序列,然后使用 Naive-Bayes、支持向量机、J48 和 AdaBoost(J48)训练分类模型,实现未知恶意软件的检测. Raff[9]等人提出省去特征工程,将整个恶意软件原生字节序列作为输入,通过神经网络去自主学习其中具有判别力的特征信息. Nataraj[10]等将恶意软件的二进制文件转化为灰度图像,使用图像处理的方式进行恶意软件分类. 机器学习的方法在一定程度上缓解了传统方法在人力和时间成本上的巨大压力,但在面对不平衡的已知和未知标签的数据集时显得无所适从,以及在大规模数据集上的表现不尽如人意.

在自然语言处理领域,用神经概率语言模型(NPLM)来量化表示单词已经有很长一段历史. 它是一种基于神经网络的语言模型,具有模拟高维离散分布的能力,其概念基于分布式假设,通过使用神经网络从词共现中提取语言的相似性和语义信息,从而产生低维、固定长度的数字向量(也称神经文本嵌入). 神经概率语言模型催生了许多优秀的算法,其中 word2vec[11]及其扩展 doc2vec[12,13]可以说是近年来最突出的代表,利用更简单和更高效的神经架构来将词或文本表征为实数值向量的一种高效的算法模型,这种向量表示方法既能表示词本身又能表达词之间的语义和语法信息,在各种领域都展现了其极高的价值,包括机器翻译、情感分析等. 而且 Word2vec 大受欢迎的还有一个原因正是其高效性,Mikolov[11]在论文中指出,一个优化的单机版本一天可训练上千亿词. 因此,本文着重研究并分析了自然语言处理在文本分类及情感分析方面的优势[14],将每个恶意软件看成是一个文本,文本的内容由恶意软件动态执行时的 API 序列构成,不同于以往的工作[15],我们并没有对 API 进行粗略的提取以及归类. 相反,为了保持行为语义的完整性,我们着重分析了 API 在进程和线程中的调用过程,保留其先后顺序,构成一个连续且上下文语义丰富的文本,通过神经概率语言模型学习文本的向量表示来量化表示恶意软件,最终用于各种恶意软件的分析任务.

3 本文方法

3.1 系统概述

本文设计并实现了一套完整的自动化恶意软件家族分类方案,该方案流程如图 1 所示,包括了特征提取、训练模型、特征表示和分类模型四大组件. 首先我们从恶意软件样本中提取 API 序列,构成恶意软件样本的文本表示,然后将所有这些文本通过一个神经网络模型进行训练学习,模型的输出

是每个恶意样本的文本嵌入表示形式，这些文本的嵌入表示保留了原始文本的语义信息，可直接用于分类模型的输入或者其它恶意软件相关问题的研究.

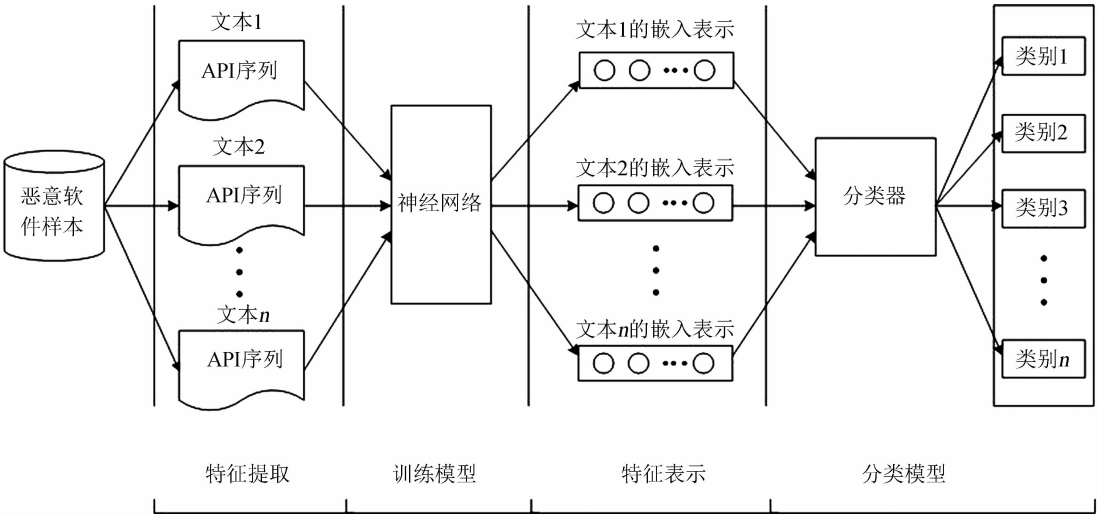

图 1　系统架构

Fig. 1　System architecture

3.2　数据集

本文所用恶意软件样本来自 VX-Heaven 网站，该网站包含了从各种来源收集到的恶意软件样本，包括蜜罐、垃圾邮件、反病毒供应商和安全研究人员，所有这些样本都带有按卡巴斯基命名规则命名的家族标签. 其中各家族样本的数量参差不齐，最少的家族仅有一个样本，最多的则有 1183 个样本，为了保持样本数据集的均衡，我们将样本数量少于 100 个的家族舍去，最终保留了 52 个家族、20386 个样本用于我们的实验. 我们利用 3.3 节介绍的方法提取该数据集的 API 序列，构建恶意软件的语料库. 通过进一步统计分析发现，整个语料库的大小为 223，表示共有 223 个不同 API. 另外，这些文本的长度范围在 1～299606 之间，说明其中最小的文本仅由 1 个 API 构成，平均文本长度为 2940，我们对文本长度低于平均长度的文本进行可视化，其分布如图 2 所示，从图中可以看出绝大多数文本的长度介于 0～1000 之间.

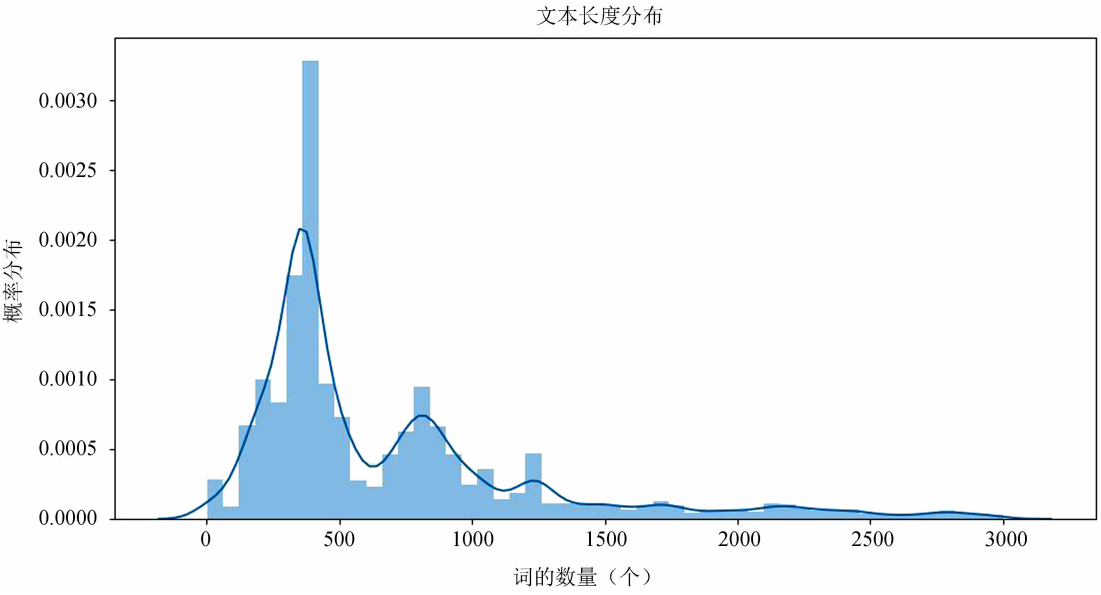

图 2　恶意软件文本长度分布

Fig. 2　The text length distribution of the malware

3.3 恶意软件文本集构建

系统调用(Application Programming Interface,API)通常作为接口提供给用户用于访问操作系统的功能和系统资源,且因其良好的规范性,不易被恶意软件作者修改,故可以使用系统调用来表征一个软件的行为. 本文利用 Cuckoo Sandbox (https://cuckoosandbox.org/)对恶意软件执行过程进行监控和记录,Cuckoo Sandbox 将返回一个 json 格式的日志,该日志记录了恶意软件执行过程中进程与线程的运行情况,为了构建一个恶意样本集的语料库,我们从日志中抽取出系统调用 API,且按其执行流的调用顺序构成 API 序列,因为从直观上来讲一段连续的 API 调用序列反映了一段具有连贯语义的行为信息. 而且对提取到的 API,我们并没有做过多的处理,仅是在提取过程中去掉了出现在同一个 API 中的 A、W 后缀,如 CreateMutexA 和 CreateMutexW 其实是代表同一个 API 调用. 最终,所有的恶意样本构成一个文本集,文本集的每一行代表一个样本,由样本名字和对应的 API 序列构成,文本集的行数即为样本集的大小,该文本集直接用于接下来神经网络模型的输入数据.

3.4 恶意软件文本的嵌入表示

Google 于 2013 年推出了一款用于训练词向量的工具 word2vec(https://code.google.com/archive/p/word2vec/),顾名思义,该工具的特点是将所有的词向量化,这样词与词之间就可以定量的去度量他们之间的关系,挖掘词之间的联系. word2vec 输出的词向量可以被用来做很多 NLP 相关的应用工作,比如聚类、找同义词、词性分析等. 然而,有时候我们需要得到文本的向量表示,理论上虽然可以直接对单词的向量取均值作为文本的向量表示,但是这样会忽略了单词之间的排列顺序对句子或文本信息的影响. 基于此,Le[12]等人提出了其扩展模型 doc2vec,doc2vec 模型实质是在 word2vec 模型的基础上做了一点改进,除了增加一个段落向量以外,这个方法几乎等同于 word2vec. 与传统文本向量空间模型相比,使用分布式向量模型表示文本,既能解决传统向量空间模型的高维稀疏特征问题,还能引入传统模型不具有的语义特征,有利于文本的分类. 如果换个思路,把 API 当作词,API 序列当作句子,多个句子可以构成一篇文本,那么 doc2vec 就可以把恶意软件映射到 K 维向量空间,从而将 API 之间的关系挖掘转化为向量空间的定量分析.

3.4.1 word2vec word2vec 是一个简化的神经网络模型,仅拥有输入层-隐藏层-输出层三层结构,算法使用浅层神经网络在多维空间中生成词的分布式数值向量表示,其目标是在给定文本语料库的情况下找到词嵌入,换句话说,这是一种用于查找词的低维表示的技术. 实际上 word2vec 使用了两个主要模型,连续词袋模型(Continuous Bag-of-Words Model, CBOW)和 Skip-Gram(SG). CBOW 的目标是根据上下文来预测当前词的概率. SG 刚好相反,根据当前词来预测上下文的概率. 由于 CBOW 和 SG 遵循相同的原则,只是实现过程相反,因此我们着重介绍 CBOW 算法的实现过程. 如图 3 所示,在该模型中,输入层和输出层都由一个维度为 V 的 one-hot 编码向量来表示,V 代表词汇表的大小. 隐藏层用一个维度为 N 的向量表示. N 代表我们选择用来表示词嵌入的维度. 此外,该模型还包括两个权重矩阵 W 和 W',分别位于输入层和隐藏层之间,以及隐藏层和输出层之间. 其中,输入层的元素 x_k 与隐藏层的元素 h_i 之间的关系由权重 W_{ki} 表示,同样的,隐藏层元素 h_i 和输出层元素 y_j 之间的连接由元素 W'_{ij} 表示. 最终输出向量 y 需要和预期目标$\hat{y}$进行比较,如果 y 离$\hat{y}$越近,说明该神经网络表现越好. 该模型具体训练过程如下.

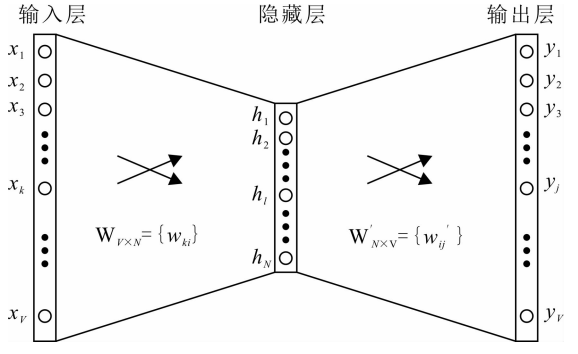

图 3 CBOW 模型

Fig. 3 CBOW Model of word2vec

1) 随机的初始化输入/输出的权重矩阵 W 和 W';

2) 预测目标词的向量$\hat{y}$;

$$h=W^T x \tag{1}$$

$$u_j=W'^T h \tag{2}$$

$$p(w_j|w_I)=\hat{y}=\frac{\exp(u_j)}{\sum_{j'=1}^{V}\exp(u_{j'})} \tag{3}$$

3）通过具有反向传播的随机梯度下降等优化算法更新权重矩阵，以找到最优的 W 和 W' 来最小化损失函数.

$$\min(L)=-\log y_{j*}=$$
$$-\log\left(\frac{\exp(u_{j*})}{\sum_{j'=1}^{V}\exp(u_{j'})}\right)=$$
$$-u_{j*}+\log\sum_{j'=1}^{V}\exp(u_{j'}) \tag{4}$$

4）最终取隐藏层和输出层之间的权重矩阵 W' 用来作为每个词的向量表示.

3.4.2　doc2vec　本质上，doc2vec 就是 word2vec 的扩展模型，通过在输入层增加一个文本 ID 来学习文本的向量表示，因此该模型适用于可变长度的文本. 与 word2vec 一样，doc2vec 也有两种模型，分别为：Distributed Memory(DM)和 Distributed Bag of Words(DBOW). DM 试图在给定上下文和文本向量的情况下预测单词的概率. DBOW 则在仅给定文本向量的情况下预测文本中一组随机单词的概率. 相比于 word2vec，其训练过程主要有以下区别.

1）训练过程中新增了文本 ID，首先将每个文本 ID 和语料库中的所有词初始化一个 K 维的向量，然后将文本向量和上下文词的向量输入模型，隐藏层将这些向量累加(或取均值、或直接拼接起来)得到中间向量，作为输出层 softmax 的输入. 在一个文本的训练过程中，文本 ID 保持不变，共享着同一个文本向量，相当于每次在预测单词的概率时，都利用了整个句子的语义.

2）在预测阶段，给待预测的句子新分配一个 ID，词向量和输出层 softmax 的参数保持训练阶段得到的参数不变，重新利用梯度下降训练待预测的句子. 待收敛后，即得到待预测句子的向量表示.

3.4.3　mal2vec　基于以上两种模型，我们提出了一种将恶意软件量化表示的模型 mal2vec. 该模型的训练数据为 3.3 节里面构建的文本集，文本集的一行表示一个恶意软件样本，由样本名字和 API 序列构成. 对于 word2vec 模型来说，因其不需要学习文本的向量表示，故其输入仅为 API 序列表示的特征矩阵，通过训练，最终每个 API 都用一个低维的向量表示，也称词嵌入. 而对于 doc2vec，因其模型增加了对文本的向量表示，故其输入为样本名字和 API 序列共同组成的特征矩阵，通过训练，最终会得到训练样本中所有 API 的向量和每个样本的向量，也即文本的嵌入表示.

4　实验结果与分析

4.1　模型参数调优

由 3.4 节原理可知，影响本文方法训练模型的参数主要有三个：vector_size(特征向量的维度)、window(上下文窗口大小)和 d_m(训练算法 DM/DBOW). 利用 3.2 节描述的数据集进行参数选择，实验结果如下.

表 1　模型参数的调优

Tab. 1　Optimization of model parameters

vector_size	window	d_m	F_1
10	5	0	0.91
20	5	0	0.95
50	5	0	0.96
100	5	0	0.95
150	5	0	0.94
50	5	1	0.90
100	5	1	0.91
150	5	1	0.90
50	1	0	0.96
50	2	0	0.96
50	3	0	0.96
50	4	0	0.96
50	5	0	0.96
50	6	0	0.96
50	7	0	0.96
50	8	0	0.96
50	9	0	0.96

表 2　分类表现评估指标

Tab. 2　Evaluation indices of classification performance

评估指标	描述
TP	将相似的两个恶意软件归类到同一家族的样本数量
TN	将不相似的两个恶意软件归类到不同家族的样本数量
FP	将不相似的两个恶意软件误归类到相同家族的样本数量
FN	将相似的两个恶意软件误归类到不同家族的样本数量
Precision	$TP/(TP+FP)$
Recall	$TP/(TP+FN)$
ACC	$(TP+TN)/(TP+TN+FP+FN)$
F_1	$2*\mathrm{Precision}*\mathrm{Recall}/(\mathrm{Precision}+\mathrm{Recall})$

如表 1 所示，我们首先将窗口大小 window 设置为默认值 5，然后分别对两种训练算法 $d_m=0$(DBOW)和 $d_m=1$(DM)都采用三个不同的维度 50，100 和 150 进行实验，通过实验结果我们发现

$d_m=0$ 的分类效果更好,从而在后面的实验中我们选择训练算法 DBOW,即 $d_m=0$,接着我们控制 window=5 和 $d_m=0$,对特征向量维度进行变化,发现当维度为 50 时效果最好,因此我们控制向量维度 vector_size=50 和 $d_m=0$,对窗口大小进行变化,发现窗口大小对分类效果几乎没有太大影响,因为模型在训练过程中是随机选择[1, window]区间的值进行训练,所以最终我们的模型参数设置为 vector_size=50,window=5 以及 $d_m=0$.

4.2 分类表现评估

我们将数据集划分成了训练集和测试集两个部分用于分类实验,其中训练集样本占样本总量的 80%,测试集占 20%. 此外我们还采用了如表 2 所示的几种常用分类效果评估指标来对基于文本嵌入表示的恶意软件进行分类表现评估. 我们通过对分类结果构建混淆矩阵,从而可以计算各家族的相应指标,最终分类表现如表 3 所示.

表 3 基于文本嵌入特征表示方法的分类表现

Tab. 3 Classification performance based on text embedding feature representation

家族	F1	ACC	TP	TN	FP	FN
Backdoor. Win32. BlackHole	0. 97	1. 00	77	3997	2	2
Backdoor. Win32. Ceckno	0. 94	1. 00	65	4005	5	3
Backdoor. Win32. Delf	0. 96	1. 00	174	3888	14	2
Backdoor. Win32. Frauder	0. 92	1. 00	18	4057	1	2
Backdoor. Win32. Hupigon	0. 98	1. 00	164	3907	5	2
Backdoor. Win32. IRCBot	0. 93	1. 00	57	4013	1	7
Backdoor. Win32. PcClient	0. 97	1. 00	29	4047	2	0
Backdoor. Win32. Poison	0. 98	1. 00	162	3909	3	4
Backdoor. Win32. Popwin	0. 99	1. 00	126	3950	2	0
Backdoor. Win32. SdBot	0. 92	1. 00	95	3967	15	1
Backdoor. Win32. Shark	0. 95	1. 00	53	4019	4	2
Backdoor. Win32. Sinowal	1. 00	1. 00	22	4056	0	0
Email-Worm. Win32. LoveLetter	0. 99	1. 00	35	4042	1	0
Email-Worm. Win32. Zhelatin	0. 91	1. 00	20	4054	2	2
HackTool. Win32. VB	0. 97	1. 00	19	4058	0	1
Hoax. Win32. Bravia	0. 87	1. 00	20	4052	2	4
Hoax. Win32. Renos	0. 85	1. 00	23	4047	6	2
Net-Worm. Win32. Kolabc	0. 94	1. 00	30	4044	1	3
Trojan-Banker. Win32. Banbra	0. 91	1. 00	91	3970	10	7
Trojan-Banker. Win32. Banker	0. 89	0. 99	98	3956	16	8
Trojan-Clicker. Win32. Delf	0. 89	1. 00	42	4026	9	1
Trojan-Downloader. Win32. VB	0. 99	1. 00	143	3932	3	0
Trojan-Downloader. Win32. Winlagons	0. 91	1. 00	39	4031	1	7
Trojan-Downloader. Win32. Zlob	0. 99	1. 00	132	3944	1	1
Trojan-Dropper. Win32. KGen	0. 98	1. 00	28	4049	0	1
Trojan-GameThief. Win32. Lmir	0. 88	1. 00	39	4028	2	9
Trojan-GameThief. Win32. Nilage	0. 93	1. 00	62	4007	1	8
Trojan-PSW. Win32. Agent	0. 84	1. 00	21	4049	1	7
Trojan-PSW. Win32. LdPinch	0. 99	1. 00	128	3948	0	2
Trojan-Proxy. Win32. Puma	0. 96	1. 00	39	4036	1	2
Trojan-Spy. Win32. Agent	0. 83	1. 00	32	4033	6	7

(续表 3)

家族	F1	ACC	TP	TN	FP	FN
Trojan-Spy. Win32. BZub	0.99	1.00	35	4042	0	1
Trojan-Spy. Win32. Banbra	1.00	1.00	20	4058	0	0
Trojan-Spy. Win32. Banker	0.94	1.00	31	4043	3	1
Trojan-Spy. Win32. Delf	0.90	0.99	98	3958	7	15
Trojan-Spy. Win32. Pophot	1.00	1.00	149	3928	0	1
Trojan-Spy. Win32. Zbot	1.00	1.00	126	3952	0	0
Trojan. Win32. DNSChanger	0.97	1.00	86	3986	0	6
Trojan. Win32. Dialer	0.99	1.00	101	3974	2	1
Trojan. Win32. Diamin	1.00	1.00	22	4056	0	0
Trojan. Win32. FraudPack	0.99	1.00	51	4026	1	0
Trojan. Win32. Inject	0.99	1.00	160	3914	1	3
Trojan. Win32. Midgare	0.99	1.00	172	3902	1	3
Trojan. Win32. Monder	0.99	1.00	237	3838	3	0
Trojan. Win32. Monderb	1.00	1.00	404	3670	4	0
Trojan. Win32. Pakes	0.90	1.00	81	3979	3	15
Trojan. Win32. Radi	1.00	1.00	31	4047	0	0
Trojan. Win32. Slefdel	0.95	1.00	52	4020	6	0
Trojan. Win32. VB	0.97	1.00	79	3994	3	2
Trojan. Win32. Vaklik	0.99	1.00	90	3986	0	2
Worm. Win32. Downloader	1.00	1.00	20	4058	0	0
Worm. Win32. Runfer	0.91	1.00	36	4035	3	4
Backdoor. Win32. BlackHole	0.97	1.00	77	3997	2	2
Backdoor. Win32. Ceckno	0.94	1.00	65	4005	5	3
avg / total	0.96	1.00	4164	207587	154	151

从表 3 可以看出，本文方法分类准确率达到 100%，F_1 分数也达到了 96%，表明基于文本嵌入的特征表示方法能够对恶意软件家族行为信息进行有效捕捉. 特别的，我们发现其中很多家族的分类指标都达到了 99%以上，说明其家族的行为特征具有明显判别力，也反映了其行为语义上的相似，证明了 API 序列表征恶意软件行为的有效性.

此外，我们还对 Rieck[1] 等人提出的方法 malheur (https://github.com/rieck/malheur)进行了复现，用于与本文方法 mal2vec 进行对比. 对比的方法是采用其公开数据集 Malheur Dataset (https://www.sec.cs.tu-bs.de/data/malheur/) 作为 malheur 和 mal2vec 的实验数据集，该数据集包含了两个部分：参考数据集和应用数据集，且每个数据集都由三种不同格式的恶意软件行为报告构成：xml 格式、CWSandbox 顺序版本格式和 MIST 格式. mal2vec 从 xml 格式的行为报告中提取系统调用序列进行训练学习，malheur 则支持 xml 和 MIST 两种格式. 其分类效果如表 4 所示.

表 4　本文方法 mal2vec 与 malheur 方法对比

Tab. 4　Compared with the malheur method

方法	数据集/格式	Precision	Recall	F_1
malheur	Reference/xml	1.00	0.67	0.80
malheur	Reference/mist	0.99	0.99	0.99
mal2vec	Reference/xml	1.00	0.99	0.99
malheur	Application/mist	0.83	0.46	0.59
malheur	Application/xml	1.00	0.42	0.59
mal2vec	Application/xml	0.87	0.89	0.88

从表 4 可以看出，对于仅有 3131 个少量样本的参考数据集而言，mal2vec 与 malheur 基于 mist 指令格式的方法分类效果相当，因此 malheur 生成 mist 指令的步骤显得多余，且可读性差. 而对于拥有 33685 个样本的应用数据集，除去 7612 个未知

家族标签的样本,再筛选样本个数大于 100 个的家族,最终剩下 22757 个样本,26 个家族用于对比试验,实验结果如表 4 所示,本文方法分类效果明显优于 malheur.

4.3 可视化展示

本文对 3.2 节数据集抽取出 10000 个样本,13 个家族利用 t-SNE 技术进行可视化展示,效果如图 4 所示,图中右上角的数字代表 13 个不同的家族,我们可以看出同一家族的样本呈聚集的态势,不同家族的样本有明显的间距,这再一次验证了本文方法量化表征恶意软件的有效性. 此外,我们还挑选出了与每个家族最相关的前 10 个 API 作为关键词,也就是图中所示每个家族上面悬浮的字符串. 我们还可以通过不同的定量分析来发现更多有关恶意软件家族行为的有趣信息.

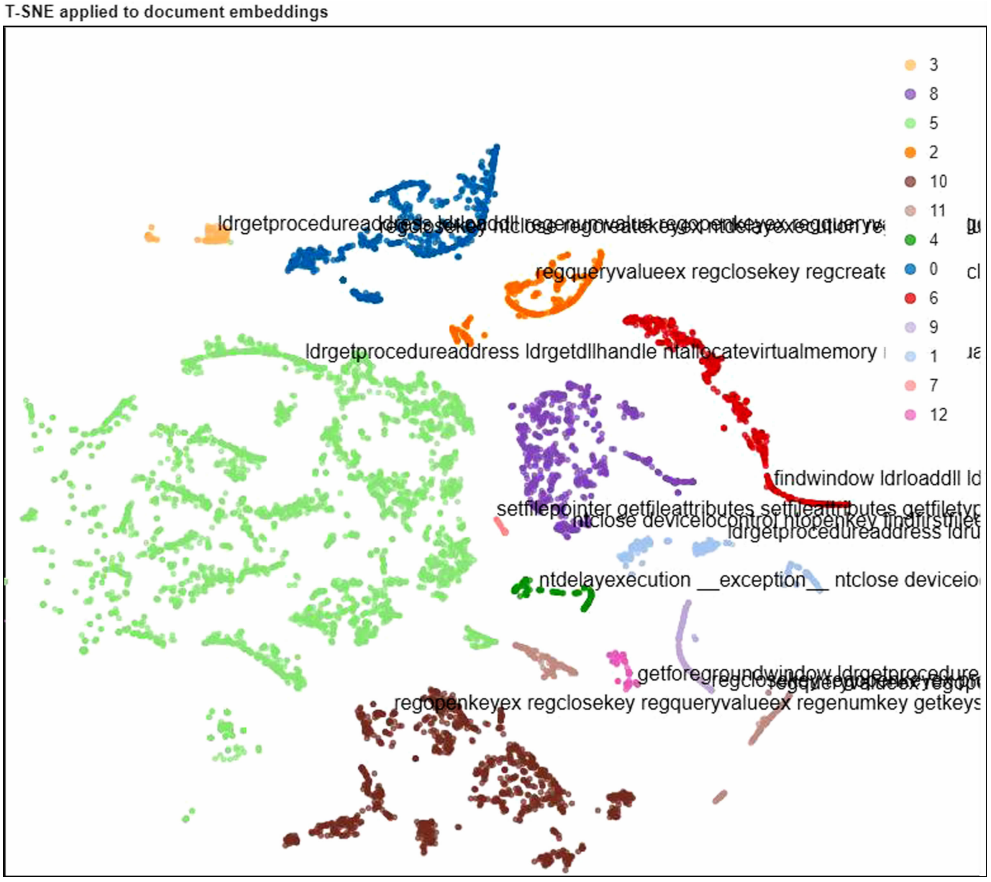

图 4 基于文本嵌入特征表示的可视化

Fig. 4 Visualization based on text embeddings

5 结 论

本文深入研究并分析了传统恶意软件分析方法在量化表示恶意软件方面的不足,发现恶意软件分析与文本分析本质上的关联,结合当前主流 word2vec 模型在自然语言处理领域的突出表现,提出一种基于神经概率模型的方法来对恶意软件进行量化表示,通过实验验证了本文方法表征恶意软件行为特征的有效性. 在与 malheur 的对比试验中发现,其生成 mist 指令的步骤显得多余,且可读性差,而本文方法省去了不必要的特征工程,仅用更简单可读的 API 特征就达到了更好的分类效果,且通过可视化实验展示了本文方法能够有效的学习到恶意软件行为语义的信息,该信息对于恶意软件分析的其他应用场景如分类、家族行为模式的分析都具有重要意义. 在未来的工作中,我们将进一步改进本文训练模型,让其能够适应变长序列,这种变长序列能够让我们对恶意软件家族的行为模式有一个直观的感受,相当于构建一个家族的 DNA.

参考文献:

[1] Rieck K, Trinius P, Willems C, *et al*. Automatic analysis of malware behavior using machine learning

[J]. J Comput Secur，2011，19：639.

[2] 肖锦琦，王俊峰. 基于模糊哈希特征表示的恶意软件聚类方法 [J]. 四川大学学报：自然科学版，2018，55：469.

[3] Karampatziakis N，Stokes J W，Thomas A，*et al*. Using file relationships in malware classification [C]//Proceedings of the 9th international Conference on Detection of Intrusions and Malware，and Vulnerability Assessment. Heraklion，Crete，Greece：Springer，2013.

[4] Dahl G E，Stokes J W，Deng L，*et al*. Large-scale malware classification using random projections and neural networks [C]//Proceedings of the International Conference on Acoustics，Speech and Signal Processing. Vancouver，BC，Canada：IEEE，2013.

[5] Kolter J Z，Maloof M A. Learning to detect and classify malicious executables in the wild [J]. J Mach Learn Res，2006，7：2721.

[6] Schultz M G，Eskin E，Zadok F，*et al*. Data mining methods for detection of new malicious executables [C] //Proceedings of the 2001 IEEE Symposium on Security and Privacy. S&P 2001. Oakland，CA，USA，USA：IEEE，2001.

[7] Kolter J Z，Maloof M A. Learning to detect malicious executables in the wild [C]//Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining. Seattle，WA，USA：ACM，2004.

[8] 魏琴芳，李林乐，张峰，等. 一种安卓系统手机恶意软件链接串行联合检测方法 [J]. 重庆邮电大学学报：自然科学版，2017，29：6.

[9] Raff E，Barker J，Sylvester J，*et al*. Malware detection by eating a whole exe [J]. Comput Res Repo，2017，2017：1710.

[10] Nataraj L，Yegneswaran V，Porras P，*et al*. A comparative assessment of mMalware classification using binary texture analysis and dynamic analysis [C]//Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence. Chicago，Illinois，USA：ACM，2011.

[11] Mikolov T，Sutskever I，Chen K，*et al*. Distributed representations of words and phrases and their compositionality [C]//Proceedings of the 26th International Conference on Neural Information Processing Systems. Lake Tahoe，Nevada：Curran Associates Inc，2013.

[12] Le Q，Mikolov T. Distributed representations of sentences and documents [C]//Proceedings of the 31st International Conference on Machine Learning. Beijing，China：JMLR. org，2014.

[13] 郭文，王俊峰. Windows 恶意代码动态通用脱壳方法研究 [J]. 四川大学学报：自然科学版，2018，55：283.

[14] Campr M，Ježek K. Comparing semantic models for evaluating automatic document summarization [C]//Proceedings of the International Conference on Text，Speech，and Dialogue. [s. l.]：Springer，Cham，2015.

[15] Liang G，Pang J，Dai C. A Behavior-Based Malware Variant Classification Technique [J]. Int J Inform Educ Tech，2016，6：291.

引用本文格式：

中　文：张涛，王俊峰. 基于文本嵌入特征表示的恶意软件家族分类 [J]. 四川大学学报：自然科学版，2019，56：441.

英　文：Zhang T，Wang J F. Malware family classification based on text embedding feature representation [J]. J Sichuan Univ：Nat Sci Ed，2019，56：441.