计算机科学
COMPUTER SCIENCE

基于FlexUDA模型的SQL注入检测研究

王清宇，王海瑞，朱贵富，孟顺建

引用本文

王清宇，王海瑞，朱贵富，孟顺建.基于FlexUDA模型的SQL注入检测研究[J]．计算机科学, 2023, 50(6A): 220600172-6.

WANG Qingyu, WANG Hairui, ZHU Guifu, MENG Shunjian. Study on SQL Injection Detection Based on FlexUDA Model [J]. Computer Science, 2023, 50(6A): 220600172-6.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220600172

基于 FlexUDA 模型的 SQL 注入检测研究

王清宇　王海瑞　朱贵富　孟顺建
昆明理工大学信息工程与自动化学院　昆明 650500
(2078125631@qq.com)

摘　要　针对深度学习方法检测 SQL 注入时有标签数据不足容易导致模型过拟合的问题,提出了一种基于半监督学习的 FlexUDA 模型。首先对采集到的数据进行解码、泛化和分词等预处理,然后通过计算 TF-IDF 值对无标签数据进行增强,并将原始数据和增强后的数据使用 TF-IDF 和 Word2Vec 融合算法进行向量化,最后使用 FlexUDA 模型进行训练,并将训练好的模型与其他模型进行对比分析。实验结果表明,FlexUDA 模型仅使用 1 000 条有标签数据和 100 000 条无标签数据进行训练,就获得了 99.42%的准确率和 99.23%的召回率,相比其他有监督训练模型,表现出了更好的泛化性能,可以很好地解决 SQL 注入检测中有标签数据不足导致的过拟合问题。

关键词:SQL 注入检测;半监督学习;无监督数据增强;动态阈值
中图法分类号　TP393.08;TP181

Study on SQL Injection Detection Based on FlexUDA Model

WANG Qingyu,WANG Hairui,ZHU Guifu and MENG Shunjian
Faculty of Information Engineering and Automation,Kunming University of Science and Technology,Kunming 650500,China

Abstract　FlexUDA model based on semi-supervised learning is proposed to solve the problem that insufficient labeled data is easy to cause model over fitting when deep learning method detects SQL injection. Firstly,the collected data are preprocessed by decoding,generalization and word segmentation,and then the unlabeled data are augmented by calculating the TF-IDF value. The original data and augmented data are vectorized using TF-IDF and Word2Vec fusion algorithm. Finally,the FlexUDA model is used for training,and the trained model is compared with other models. Experimental results show the FlexUDA model only uses 1000 labeled data and 100000 unlabeled data for training,and achieves 99.42% accuracy and 99.23% recall. Compared with other supervised training models,it shows better generalization performance,and can well solve the over fitting problem caused by insufficient labeled data in SQL injection detection.

Keywords　SQL injection detection,Semi-supervised learning,Unsupervised data augmentations,Dynamic threshold

1　引言

随着互联网技术的飞速发展,网络攻击事件愈发频繁。2013－2021 年的开放 Web 安全项目组织 OWASP TOP10 报告结果显示,注入攻击始终位列排行榜前三[1-2]。而 SQL 注入是注入攻击中最常见也是危害性最高的一种攻击类型。国内外学者针对 SQL 注入已提出了很多的检测方法,大致可分为传统方法和基于机器学习的检测方法[3]。传统方法又可分为静态方法、动态方法和动静结合的检测方法。其中静态检测方法指在程序运行前,通过对源码进行分析来判断是否存在 SQL 注入的风险[4]。例如 Livshits 等[5]建立了一种可扩展且精确的静态分析器,可以通过漏洞说明自动进行规则匹配,然后把静态分析的结果提交给用户评估。动态检测方法指在程序运行时,通过分析渗透测试、生成模型的数据流等方法来检测是否可能发生 SQL 注入[6]。例如 Das 等[7]提供了一种基于动态查询匹配的 SQL 注入检测方法,通过解析将 SQL 查询转换成 XML 形式,并与预先定义的 XML 文件进行比较,来判断是否存在注入语句。动静结合的方法旨在结合静态方法与动态方法两者的优势,在静态分析阶段,构造安全的 SQL 语句模型,然后在动态分析阶段,解释提交的 SQL 语句,将其与静态 SQL 语句模型进行对比,判断是否有 SQL 注入风险。例如 Halfond 等[8]提出的 AMNESIA,先通过分析 web 应用程序生成安全的 SQL 查询模型,然后动态监视提交的查询,若查询与静态模型不符,则判定为 SQL 注入攻击,并向开发人员或者管理员报告相关信息。传统方法,无论是静态方法、动态方法,还是将两者结合的方法,在面对超级流量的大数据时代,都存在检测效率低、成本高、占用资源多、漏报率和误报率高以及普适性差等各种各样的问题[9-11]。

机器学习的发展迎合了数字时代海量数据的特点,使得这项技术飞速发展,应用前景广阔,并取得了令人惊叹的成果。但浅层机器学习用于 SQL 注入时需要非常强的专业知识和精准全面的分析能力,才能提取到 SQL 数据的有效特征,并且随着注入攻击手段的演变和提升,模型的特征提取也

基金项目:国家自然科学基金(61863016,61263023)
This work was supported by the National Natural Science Foundation of China(61863016,61263023).
通信作者:王海瑞(hrwang88@163.com)

需要不断地手动更新，以适应新出现的注入特征。得到一个好的训练模型往往需要付出很大的代价，且模型的泛化能力通常不是很好。深度学习由于省去了复杂的人工特征提取环节，有望在 SQL 注入检测领域发挥巨大作用[12]。但是网络安全领域公开的数据集较少，有限的数据集非常容易导致深度学习得到的模型过拟合，这是目前深度学习方法检测 SQL 注入最大的难题之一。本文提出一种半监督学习方法，通过无监督数据增强技术来扩展无标签数据量，利用少量的有标签数据和大量的无标签数据进行模型训练。最后的对比实验结果证明，相比其他数据增强方法或者基于有限标签数据的有监督学习方法，本文提出的半监督模型能够得到更好的检测效果，可以很好地解决深度学习检测中的过拟合问题。

2 FlexUDA 模型

2.1 TextCNN 网络

FlexUDA 模型中采用 TextCNN 网络对数据进行训练和测试。TextCNN 是用于文本处理的卷积神经网络模型，具有等变表示、稀疏交互和参数共享等特点，能够捕捉到数据中的重要局部特征，并且在存储和计算等方面的效率极大地优于传统的使用矩阵乘法的神经网络。该网络模型主要分为 4 个部分：输入层(Input layer)、卷积层(Convolution layer)、池化层(Pooling layer)和全连接层(Fully Connected layer)。输入层是 $n\times k$ 的句子矩阵，其中 n 代表句子中的单词数，k 代表每个词对应的词向量的维度。由于句子矩阵必须统一，所以需要对句子长度进行 padding，少则补 0，多则截断。词向量的输入通常有 4 种类型[13]：1)先对不同的词向量进行随机初始化，后期训练时再做调整；2)用 Word2Vec 等方法先将单词训练成词向量，后期直接使用不再进行调整；3)使用训练好的词向量进行初始化，后期训练时依然需要调整；4)将第二、三种方法相结合。文献[14]指出，使用预先训练好的词向量可以显著提升分类效果，故本文模型中使用 TF-IDF 和 Word2-Vec 融合算法将预处理后的数据训练成词向量，然后输入模型中使用。卷积层中，TextCNN 使用长宽不等的卷积核，其中宽和词矩阵的宽相等，以保证每次滑动都能取到完整的词向量。卷积核的长一般选取多个值，以便可以提取到多个不同的上下文信息[15]。池化层对卷积层得到的大量特征进行筛选聚合，这样可以减少大量运算以及防止过拟合的发生。全连接层将前面卷积池化的结果整理成一个一维向量，并使用激活函数输出各类别的概率。图 1 给出了 TextCNN 的结构展示图，为简化说明，图中输入为 7×4 的句子矩阵，分别使用了 3 种大小(3×4，4×4，5×4)的卷积核，每种大小的卷积核有两个，采用 1-max 池化，最后得到一个二分类结果。

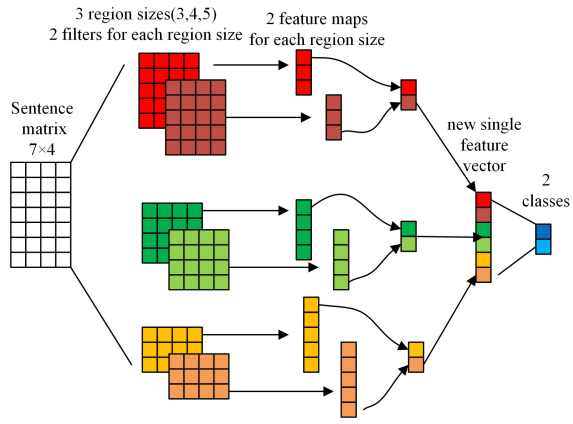

图 1 用于句子分类的 Textcnn 网络

Fig. 1 TextCNN architecture for sentence classification

2.2 无监督数据增强

无监督数据增强(UDA)是 Google 于 2019 年提出的半监督学习算法。该方法通过强制使无标签样本和增强后的无标签样本预测一致来达到更好的训练效果，从而有效解决了有标签数据集小导致的过拟合问题[16]。该模型的损失分两部分：有标签数据的交叉熵损失和无标签数据的一致性损失。其中有标签的交叉熵损失可由 $\underset{x,y^*\in L}{E}[p_\theta(y^*|x)]$ 计算得出，无标签的一致性损失可由式(1)计算得到。

$$\min_\theta \mathcal{J}_{\mathrm{UDA}}(\theta)=\mathop{\mathbb{E}}_{x\in U}\mathop{\mathbb{E}}_{\hat{x}\sim q(\hat{x}|x)}[D_{\mathrm{KL}}(p_{\tilde{\theta}}(y|x)\,\|\,p_\theta(y|\hat{x}))] \tag{1}$$

其中，x 是原始样本，$\hat{x}$ 是基于 x 的增强样本，$q(\hat{x}|x)$ 是增强变换。为了使增强变换更加有效，需要使未经增强得到的输出分布 $p_{\tilde{\theta}}(y|x)$ 和经过增强变换得到的输出分布 $p_\theta(y|\hat{x})$ 之间的差异尽可能小，即计算其 KL 散度的最小值。

为了同时使用有标签数据和无标签数据，在计算最终损失时需要将交叉熵损失加上带权重 λ 的一致性损失，最终损失公式如式(2)所示：

$$\min_\theta \mathcal{J}=\mathop{\mathbb{E}}_{x,y^*\in L}[p_\theta(y^*|x)]+\lambda\mathcal{J}_{\mathrm{UDA}}(\theta) \tag{2}$$

2.3 FlexUDA 模型结构

FlexUDA 模型结构如图 2 所示。输入是有标签数据和无标签数据。数据经过预处理后，对有标签数据使用 TextCNN 网络计算其交叉熵损失，对无标签数据进行增强，具体增强方法在 3.2.2 节中详细介绍。得到增强过的数据后便把原始无标签数据和增强后的无标签数据一起送进 TextCNN 网络中，通过 Flex 技术对无标签数据的预测值打伪标签并计算其一致性损失。最后将有监督得到的交叉熵损失和无监督得到的一致性损失按照 1:1 的比例求和，得到总损失，利用总损失回传更新网络参数，来得到一个较好的半监督训练模型。

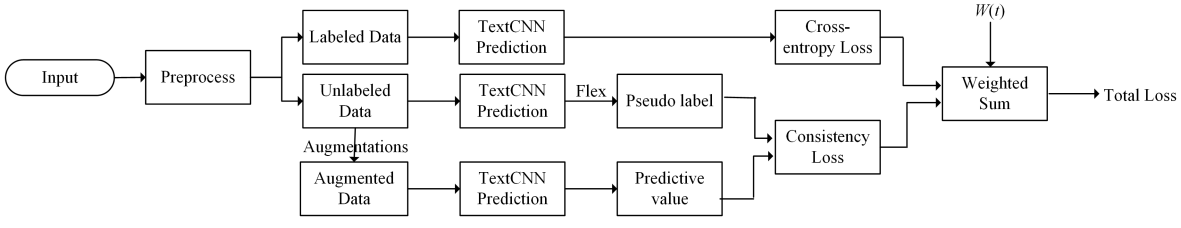

图 2 FlexUDA 模型结构

Fig. 2 FlexUDA model structure

该模型在无监督数据增强的基础上加入了 Flex 技术，即动态阈值调整。该技术是于 2021 年由 Zhang 等[17]提出的

用于优化半监督学习模型的一种优化算法。在 UDA 模型训练开始阶段，只有少数样本能够超过设定的阈值，此时每个 Batch 中被选到的样本很少，数据利用率较低，收敛速度较慢。再加上模型的随机初始化，可能使预测朝着错误的方向不断倾斜，最终导致拟合效果较差。因此本文加入 Flex 动态阈值，使阈值在开始阶段相对较低，无标签样本能够更容易地被打上伪标签，随着训练过程的进行逐步提高阈值，这样既可以加快模型的训练速度，又可以提高模型预测的准确度。假设 c 类的固定阈值为 τ，在 t 时刻的动态阈值为 $T_t(c)$，调整系数为 $\beta_t(c)$，学习效果为 $\sigma_t(c)$，则 $T_t(c)$，$\beta_t(c)$ 及 $\sigma_t(c)$ 可分别由式(3)－式(5)计算得到。

$$T_t(c)=\beta_t(c)\cdot\tau \tag{3}$$

$$\beta_t(c)=\frac{\sigma_t(c)}{\max\{\max_c \sigma_t, N-\sum_c \sigma_t\}} \tag{4}$$

$$\sigma_t(c)=\sum_{n=1}^{N}1(\max(p_{m,t}(y|u_n))>\tau)\cdot 1(\arg\max(p_{m,t}(y|u_n))=c) \tag{5}$$

式(5)中的学习效果 $\sigma_t(c)$ 指置信度高于阈值 τ 并且被正确分类到类别 c 中的样本数。式(4)中的分母是一个阈值 warm up 的过程。因为在训练的初始阶段，受模型随机初始化的影响，样本很可能被盲目地预测到其中一个类别中。分母中的第一项是一个归一化策略，代表 c 类在 t 时刻的最好学习效果，在训练初期不起作用；分母第二项表示尚未被高阈值选择过的样本数。在训练初始阶段，由于大部分样本尚未被选择到，因此分母中的第二项起主要作用。随着训练过程的进行，分母中的第一项逐渐起作用。此外还可以对调整系数 $\beta_t(c)$ 进行映射，使之非线性增长。Flex 技术提出者指出，将 $\beta_t(c)$ 映射成凸函数可以减小前期不稳定因素对阈值 τ 的影响。映射函数可以表示为式(6)。其中的 $\mathcal{M}(\beta_t(c))$ 为 $\beta_t(c)$ 的映射函数。

$$\mathcal{T}_t(c)=\mathcal{M}(\beta_t(c)) \tag{6}$$

由于训练数据由少量的有标签数据和大量的无标签数据组成，两种数据分布极不平衡，为了防止过拟合的发生，本实验采用了文献[16]中提出的 Training Signal Annealing (TSA)技术来解决该问题。这种技术通过在无标签数据增加过程中逐步去除有标签数据的方法来抑制过拟合的发生。具体来说，就是对于每一个训练步骤 t，设置一个阈值 η_t 满足 $\frac{1}{K}\leqslant\eta_t\leqslant 1$，其中 K 是类别数，本实验中 $K=2$。当有标签样本的正确分类概率 $p_\theta(y^*|x)$ 大于阈值 η_t 时，就将该标签样本从损失计算中去除，使用其他剩余标签样本继续训练。当阈值 η_t 逐步从 $\frac{1}{K}$ 变为 1 时，模型只能缓慢地从标记样本中接受监督训练，从而在很大程度上缓解了过拟合的产生。阈值 η_t 有线性变化、指数变化和对数变化等过程，3 种变化过程分别服从式(7)－式(9)，其中 T 代表总的训练步骤，t 是当前的训练步骤。本文令阈值 η_t 服从指数变化，使其在训练开始阶段增长缓慢，在训练后期增长较快，这样可以更好地抑制过拟合的产生。

$$\eta_t=\frac{t}{T}*\left(1-\frac{1}{K}\right)+\frac{1}{K} \tag{7}$$

$$\eta_t=\left(1-\exp\left(-\frac{t}{T}*5\right)\right)*\left(1-\frac{1}{K}\right)+\frac{1}{K} \tag{8}$$

$$\eta_t=\exp\left(\left(\frac{t}{T}-1\right)*5\right)*\left(1-\frac{1}{K}\right)+\frac{1}{K} \tag{9}$$

3 实验设计

本实验流程主要包括数据采集、预处理、数据增强、向量化、模型训练、模型测试六大部分。具体流程图如图 3 所示。

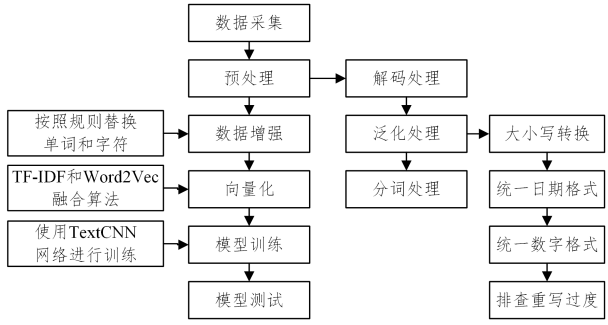

图 3 实验设计流程图

Fig. 3 Experimental design flow chart

3.1 数据采集

实验数据由四部分组成：1)从数据仓库[18]中收集并筛选有效的 SQL 注入语句和正常 SQL 语句，包括 10 000 条正样本和 20 000 条负样本；2)自己在靶场网站上运行 SQLmap 及 tamper 脚本，使用 wireshark 进行抓包获取的注入及非注入无标签数据，共计 50 000 条；3)使用本文 3.2 节提出的数据增强方法对无标签数据进行增强，得到 50 000 条无标签数据；4)按照文献[3]中方法对注入样本进行增强，生成 10 000 条正样本，此部分数据用于对比实验中。数据集总共包括 40 000 条有标签数据，正负样本比例为 1∶1，以及 100 000 条无标签数据。

3.2 数据预处理

数据预处理是 SQL 注入检测非常关键的步骤，因为数据决定了机器学习和深度学习的上限。一个优质的预处理可以排除干扰噪声和无关特征，提高训练的效率和精确度。预处理阶段主要包括解码处理、泛化处理、分词处理 3 个步骤。下面分别详细阐述。

Step1 解码处理。解码主要是针对 ASCII 编码、URL 编码、UNICODE 编码和 JSON 编码等常见编码。解码操作可以让深度学习能够学到有效的 SQL 样本特征。

Step2 泛化处理。泛化处理是为了去除相似无用的特征，让深度学习学到更加具有针对性的特征。比如 SQL 语句中是不区分大小写的，因此将所有英文字母转换成小写字母可以减少很多无用特征，同时排除掉攻击者使用的大小写混写的绕过手段。还有不同日期和数字，不会产生不同的注入影响。比如“or 1＝1＃和 or 2＝2＃”本质上是相同的，不会因为数字的不同而产生不同类型注入。排查重写过度是为了防止关键字中嵌入关键字的绕过技术，比如 SESELECTLESELECTCSELECTT 排查掉中间的几个 SELECT 之后，剩余字符恰好构成一个关键字 SELECT。为防止此类绕过方式，需要事先存储关键字，对冗余字符串进行排查。

Step3 分词处理。分词处理是为了方便后面的文本向量化。为了保留 SQL 语句中的关键字符和关键词，本文不采用 Joshi 等[19]的空白分割法，而是将所有字符包括空格在内全部保留，其中一些特殊字符如“--”可以起到单行注释作用，如果拆成两个“-”就会失去意义，因此要注意保留特殊字符的原始意义。

下面给出3个具体示例来展示原始SQL样本及预处理过的样本特征。

处理前：

(1)uNion selECt username,password from users--

(2)Select user name from databas&#101

(3)select password from tablename where id=1

处理后：

(1)['union',' ','select',' ','username',',',' ','password',' ','from',' ','users','--','\\n']

(2)['select',' ','user',' ','name',' ','from',' ','database','\\n']

(3)['select',' ','password',' ','from',' ','tablename',' ','where',' ','id','=','0','\\n']

3.3 数据增强

通过计算TF-IDF值对无标签数据进行增强变换。具体实现如下。

Step 1 首先将tfidf稀疏压缩矩阵按照tfidf值降序排列存储，并设定第500个tfidf值为阈值σ；

Step 2 循环遍历每条输入样本(一条预处理过的SQL语句)。

(1)如果某分词tfidf值大于阈值σ：以0.1的概率将该分词用前500个分词中的任意一个随机替换；

(2)如果某分词tfidf值小于阈值σ：以0.9的概率将该分词用前500个分词以后的分词中的任意一个随机替换；

(3)结束循环，输出一条增强后的无标签SQL样本。

通过以上操作，每条无标签数据都可以生成一条或者多条增强后的无标签数据。根据需要，由收集到的50 000条无标签样本通过增强生成了50 000条增强过的无标签样本，无标签数据量扩展了一倍。

3.4 文本向量化

向量化是为了将预处理后的样本数据转换为可以直接输入到算法中的数值型矩阵。词频-逆文档率(TF-IDF)是一种用于信息检索与数据挖掘的常用加权技术，可以评估一个词对于一个文件集或者一个语料库中的某个文件的重要程度。但是该方法丢弃了词条与上下文之间的联系，忽略了词语之间的语义信息。Word2Vec是Google团队开发的一种高效训练词向量的算法，该算法将词语映射成一个固定长度的短向量，通过比较向量在向量空间中的距离，来判断词语之间的相似度[20]，可以保留文本的上下文关系。TF-IDF和Word2Vec在文本向量化方面各有优势，但是单独使用都不能很全面地提取文本中的重要信息，故本文将两种算法融合，既可以提取到文本中的重要信息，又可以保留文本中的上下文关系，使深度学习模型能够学到更加有用的特征。

3.5 模型训练

模型采用TextCNN网络进行训练和测试，使用pytorch框架进行代码实现，并使用CUDA和CUDNN进行计算加速，使用tensorboardx对评价指标数据进行记录保存，使用Adam优化算法更新网络参数。该算法通过动量和自适应学习率加快网络收敛速率，实现简单，计算高效，对内存需求较小，参数的更新不受梯度的伸缩变换影响，且超参数具有很好的解释性，通常无需调整或者仅需微调。使用LambdaLR scheduler对优化器的学习率进行调整。使用交叉熵损失和一致性损失分别对有监督和无监督部分进行损失计算。

3.6 模型测试

模型测试中设置了两组对比实验，一组是将本文模型和文献[3]提出的基于数据传输信道的正样本生成方法进行对比，另一组是将本文模型和另外两种有监督训练模型进行对比。分别使用准确率、精确率、召回率、F1值等指标对两组实验进行对比分析，以此来证明本文提出的FlexUDA模型相比其他数据增强方法或者有监督模型可以更好地解决深度学习检测SQL注入时出现的过拟合问题。

4 实验与分析

4.1 实验环境

本文实验在GPU环境下进行计算，使用机器学习框架Pytorch，开发平台为Pycharm，开发语言为python。具体实验环境如表1所列。

表1 实验环境

Table 1 Experimental environment

配置项	配置参数
处理器	12thGen Intel(R) Core(TM) i5-12490F 3.0GHz
操作系统	Win10 企业版 版本号 21H1
显卡	NVIDIA GeForce RTX 3060 12GB 显存
内存	32GHz
开发平台	Pycharm 11.0.13+7-b1751.19 amd64
开发语言	Python3.9
机器学习框架	Pytorch1.8.0+cu111

4.2 评价指标

深度学习需要用合适的评价指标对模型生成的分类器进行评价，从中选出最有价值的分类器。本实验使用的评价指标有准确率(Accuracy)、精确率(Precision)、召回率(Recall)和F1-score。相关参数混淆矩阵如表2所列。

表2 混淆矩阵

Table 2 Confusion matrix

语句判定	实际为注入样本	实际为正常样本
判定为注入样本	TP	FP
判定为正常样本	FN	TN

基于以上参数，得出下面的评价指标公式：

$$Acc=\frac{TP+TN}{TP+TN+FP+FN} \tag{10}$$

$$P=\frac{TP}{TP+FP} \tag{11}$$

$$R=\frac{TP}{TP+FN} \tag{12}$$

$$F1=\frac{2TP}{(FP+2TP+FN)} \tag{13}$$

4.3 实验结果及分析

首先使用1 000条有标签数据和100 000条无标签数据作为训练集对FlexUDA模型进行训练和测试，训练集中有标签正负样本比例为1∶1，100 000条无标签数据由原始的50 000个样本和使用本文方法增强后得到的50 000个样本组成，测试集由1 000个正样本和1 000个负样本组成。实验结果用准确率Acc、精确率P、召回率R和$F1$值以及Train Loss和

Test Loss 进行描述与分析，如图 4 所示。由图 4 可以看到，FlexUDA 模型得到的准确率、精确率、召回率和 F1 值分别为 99.42%，99.01%，99.23%和99.12%，4 个指标均在 99%以上。在达到 80 个 epoch 时，训练已达到一个较好的结果，从训练损失及测试损失也可以看到，模型并未出现过拟合现象。因此该结果可以很好地证明，FlexUDA 模型仅需少量的有标签数据和大量的无标签数据便能获得出色的训练效果，而且泛化性能较好，不容易出现过拟合现象。

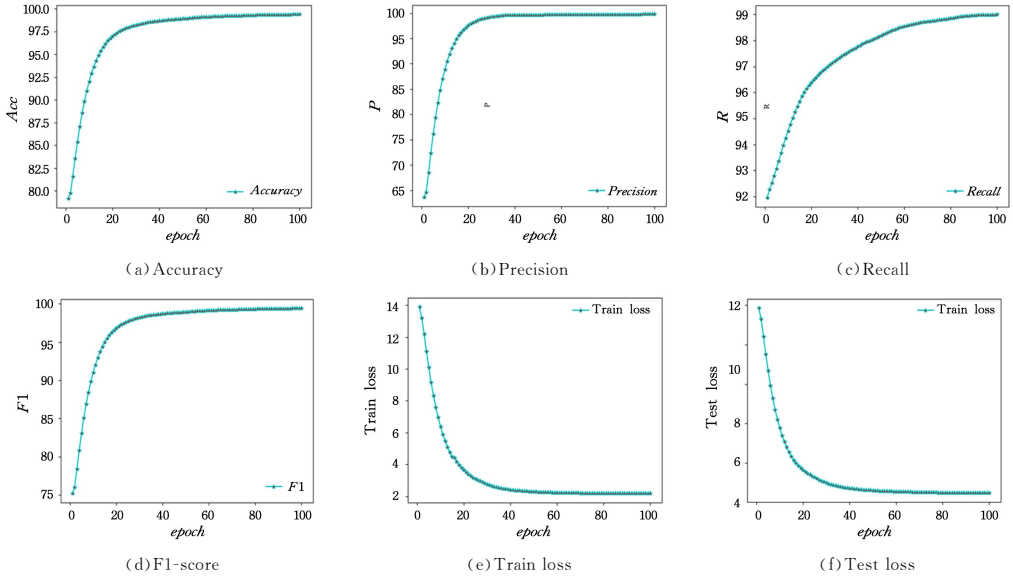

图 4　FlexUDA 模型的实验结果曲线图

Fig. 4　Experimental result curve of FlexUDA model

本文还设置了两组对比试验，其中一组是和文献[3]中的基于数据传输信道的正样本生成方法作对比，比较两种数据增强方法的效果。对比实验训练集为 10000 条原始正样本数据，10000 条生成的正样本数据，和 20000 条负样本数据，正负样本比例为1∶1。FlexUDA 模型使用 1000 条有标签数据和 100000 条无标签数据进行训练，其中有标签正负样本比例为1∶1，100000 条无标签数据包括原始的 50000 个样本和使用本文方法增强后得到的 50000 个样本。两个实验的测试集均由 1000 个正样本和 1000 个负样本组成。实验结果如表 3 和图 5 所示。

表 3　数据增强对比实验结果

Table 3　Data augmentations experiment results comparison

（单位：%）

Model	*Acc*	*P*	*R*	*F*1
文献[3]	99.52	99.15	98.72	98.93
FlexUDA	99.42	99.01	99.23	99.12

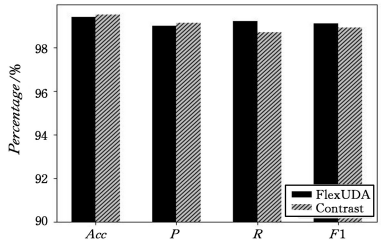

图 5　数据增强对比实验结果

Fig. 5　Data augmentations experiment results comparison

从表 3 和图 5 可以看出，本文方法得到的召回率和 F1 值相比文献[3]中基于数据传输信道的正样本生成方法分别提升了 0.51%和 0.19%，准确率和精确率略有下降。在 SQL 注入检测中，我们更注重召回率，通常都希望召回率较高一些，宁愿牺牲一部分的误报率，也不希望注入的发生，因为注入成功可能会给数据库造成巨大的损失。而 F1 值可以综合衡量召回率和精确率两个指标，常被视为分类器好与坏的综合性能指标。除此之外，本文的样本增强是针对无标签数据进行增强，相对而言有标签数据的使用量大幅缩减，只有对比实验的 1/40，但检测效果依然优于文献[3]中的针对有标签数据的增强方法。综合评估后可以看出，本文方法更胜一筹。

另一组对比实验是验证相比文献[21]中基于 LSTM-RNN 有监督训练模型和文献[22]中基于 CNN 有监督训练模型，本文所用半监督模型的泛化性能是否更为出色。两组有监督模型使用的训练集均为 40000 个有标签数据，其中包括 20000 个正样本和 20000 条负样本，正负样本比例为1∶1，模型参数参照原文献中的数据进行微调，使模型达到较好的效果。FlexUDA 模型使用 1000 个有标签数据和 100000 个无标签数据进行训练，其中有标签正负样本比例为 1∶1，100000 个无标签数据包括原始的 50000 个样本和使用本文方法增强后得到的 50000 个样本。测试集均由 1000 个正样本和 1000 个负样本组成。实验结果如表 4 和图 6 所示。由表 4 和图 6 可以看出，与有监督的 LSTM-RNN 和 CNN 模型相比，本文的 FlexUDA 模型检测效果明显好很多。FlexUDA 模型的 4 个指标均比另外两种模型高，其中的召回率和 F1 值，分别比 LSTM-RNN 和 CNN 高出 2.80%，2.25%和 1.06%，1.28%。其主要原因就在于有标签数据不够充足导致有监督模型训练过拟合，从而在测试集上表现不佳。从训练数据量来看，FlexUDA 模型仅使用有监督模型 1/40 的有标签数据，工作量大大减小，而无标签数据的获取和增强较容易，并不会增加很多额外的工作量。因此与有监督模型相比，本文 FlexUDA 模型拥有较大的优势，可以很好地解决有标签数据不足导致的过拟合问题。

表 4 与有监督模型对比的实验结果

Table 4 Comparison with supervised model

(单位:%)

Model	*Acc*	*P*	*R*	*F*1
LSTM-RNN	98.91	97.32	96.43	96.87
CNN	97.28	97.52	98.17	97.84
FlexUDA	99.42	99.01	99.23	99.12

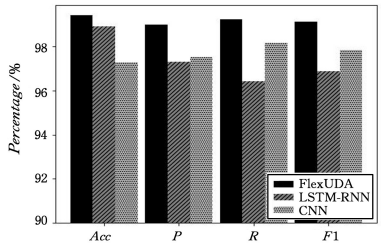

图 6 与有监督模型的对比实验结果

Fig. 6 Comparison with supervised model

结束语 本文提出了一种用于解决深度学习训练中有标签数据不足导致过拟合问题的改进半监督学习算法。首先对采集到的数据进行解码、泛化和分词等预处理,然后通过计算 TF-IDF 值对无标签数据进行增强,并将原始数据和增强后的数据使用 TF-IDF 和 Word2Vec 融合算法进行向量化,最后使用 FlexUDA 模型进行训练,并将训练好的模型与其他模型进行对比分析。实验证明了本文模型在有限标签数据用于训练的情况下,相比其他数据增强方法和有监督模型,能够更好的解决过拟合问题。如何在使用大量无标签数据提高训练效果的同时提高训练效率,是下一步要解决的问题。

参 考 文 献

[1] Top 10 Web Application Security Risks [EB/OL]. https://owasp.org/www-project-top-ten.

[2] OWASP TOP 10 from 2003 to 2021 Releases [EB/OL]. https://github.com/ OWASP/Top10.

[3] WANG F. Research and implementation of SQL injection detection technology based on d-eep learning[D]. Beijing: Beijing University of Posts and Telecommunications, 2020.

[4] GOULD C, SU Z, DEVANBU P. Static checking of dynamically generated queries in database applications[C]// 26th International Conference on Software Engineering. IEEE, 2004: 645-654.

[5] LIVSHITS V B, LAM M S. Finding Security Vulnerabilities in Java Applications with Static Analysis[C]// Proceedings of the 14th Conference on USENIX Security Symposium. 2005: 18.

[6] SHIN Y. Improving the identification of actual input manipulation vulnerabilities[C]// 14th ACM SIGSOFT Symposium on Foundations of Software Engineering ACM. 2006.

[7] DAS D, SHARMA U, BHATTACHARYYAD K. An Approach to Detection of SQL Injection Vulnerabilities Based on Dynamic Query Matching[J]. International Journal of Computer Applications, 2010, 1(25): 39-45.

[8] HALFOND W G J, ORSO A. AMNESIA: analysis and monitoring for neutralizing SQL inj-ection attacks[C]// Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering. 2005: 174-183.

[9] XIAO Z, ZHOU Z, YANG W, et al. An approach for SQL injection detection based on behavior and response analysis[C]// 2017 IEEE 9th International Conference on Communication Software and Networks(ICCSN). IEEE, 2017: 1437-1442.

[10] APPIAH B, OPOKU-MENSAH E, QIN Z. SQL injection attack detection using fingerprints and pattern matching technique [C]// 2017 8th IEEE International Conference on Software Engineering and Service Science(ICSESS). IEEE, 2017: 583-587.

[11] WASSERMANN G, GOULD C, SU Z D, et al. Static Checking of Dynamically Generated Queries in Database Applications[J]. ACM Transactions on Software Engineering and Methodology, 2007, 16(4): 14.1-14.27.

[12] ISHITAKI T, OBUKATA R, ODAT, et al. Application of deep recurrent neural networks for prediction of user behavior in tor networks[C]// 2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA). IEEE, 2017: 238-243.

[13] ZHANG X Y. Research on patriotism in class-ical poetry based on textcnn[D]. Shanghai: Shanghai Normal University, 2020.

[14] LECUN Y, BENGIO Y, HINTON G. Deep learning[J]. Nature, 2015, 521(7553): 436-444.

[15] CHEN Y. Convolutional neural networks for sentence classification[D]. Waterloo: University of Waterloo, 2015.

[16] XIE Q, DAI Z, HOVY E, et al. Unsupervised data augmentation for consistency training[J]. Advances in Neural Information Processing Systems, 2020, 33: 6256-6268.

[17] ZHANG B, WANG Y, HOU W, et al. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling[J]. Advances in Neural Information Processing Systems, 2021, 34: 18408-18419.

[18] SQL injection dataset [EB/OL]. [https://github.com/client9/libinjection.

[19] JOSHI A, GEETHA V. SQL Injection detection using machine learning[C]// 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT). IEEE, 2014: 1111-1115.

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

[21] LI C. Research on SQL injection detection technology based on Naive Bayes and LST-M recurrent neural network[D]. Changsha: Hunan University, 2018.

[22] CAO X B. Research on SQL injection detect-ion based on deep learning[D]. Nanning: Guangxi University, 2020.

WANG Qingyu, born in 1995, postgraduate. His main research interests include cyber security and machine learning.

WANG Hairui, born in 1969, Ph.D, professor, is a member of China Computer Federation. His main research interests include multimedia intelligence technology, network control technology, and embedded application technology.