第3章

网络安全中的AI应用与检测

随着网络空间安全问题向复杂化、社会化等趋势发展，在网络攻击与防御中运用人工智能技术越来越成为非常必要的选择。而许多网络空间安全问题的检测识别都可以归结为分类问题，进而运用分类器相关理论和技术来解决。本章以网络入侵、SQL 注入以及虚假信息检测为例介绍机器学习分类器技术的运用方法，涉及网络层、应用层和内容层。同时，这些例子都属于典型的对抗场景应用，是理解和学习人工智能安全的起点。

3.1 网络入侵与检测

入侵检测是网络安全中的经典问题，入侵是指攻击者违反系统安全策略，试图破坏计算资源的完整性、机密性或可用性的任何行为。由定义可见，入侵并非一种特定的入侵行为，而是一类入侵行为的统称。常见的网络攻击方式包括拒绝服务攻击、伪装身份入侵等。

3.1.1 AI 安全的起点

在网络入侵中，攻击者的目的在于突破网络安全防线，进而打通攻击途径、获得目标数据、安装后门软件等。攻击者主要利用网络层或应用层漏洞，采取灵活多样的攻击手段。

这种网络入侵行为必然要求安全管理人员采取高效的检测手段，以便能应对入侵行为。其基本要求是高效性、及时性、高准确性等。依靠人工和简单规则匹配无法满足这些要求，因此，防御者必然要使用人工智能技术。而正是人工智能技术的运用，才使得攻防双方的对抗进入新的阶段，即人工智能模型与数据安全层面。这个阶段的入侵检测可以看作人工智能安全的起点，是 AI 安全的初始对抗场景。

3.1.2 AI 应用的总体情况

入侵检测系统(Intrusion Detection System,IDS)是一种网络安全设备,可以对入侵行为进行实时监测,并在必要时发出告警或采取防御措施,切断入侵者的网络访问。最早IDS 系统的相关介绍由 Denning 于 1980 年发表于 IEEE 软件工程汇刊上。

IDS 有多种不同的划分方法,可以根据信息来源、检测方法、体系结构进行分类。根据信息来源可分为基于主机的 IDS、基于网络的 IDS 和混合型 IDS;根据检测方法可分为异常检测和误用检测;根据体系结构的不同,可以分为集中式 IDS 和分布式 IDS。以下对这些主要 IDS 模型进行介绍。

(1) 异常检测(anomaly detection):这种方法要求先建立正常行为的特征轮廓和模式表示,然后在检测时将具体行为与正常行为进行比较,如果偏差超过一定值,则认为是入侵行为,否则为正常行为。这种检测模型不需要对每种入侵行为进行定义,能有效检测未知的入侵,因此漏报率低,但误报率高。

(2) 误用检测(misuse detection):事先构建异常操作的行为特征,建立相应的模式特征库。当监测到的用户或系统行为与特征库中的记录相匹配时,则认为发现入侵。与异常检测方法相反,这种方法误报率低、漏报率高。

(3) 基于主机的 IDS:其数据来源于计算机操作系统的事件日志、应用程序的事件日志、系统调用、端口调用和安全审计记录。因此,这种 IDS 是对主机入侵行为的检测。

(4) 基于网络的 IDS:这种 IDS 用于检测整个网段的入侵信息。其数据来源于网络通信数据包,由部署于网络的数据包采集器嗅探网络上的数据包。这种数据包涵盖了各种类型网络的请求和响应记录,通常由 IP 地址、端口号、数据包长度等信息组成。

(5) 混合型 IDS:前述各种 IDS 都存在一定不足,各有其优势和缺点,因此混合型 IDS 能够较好地整合各自的优势。混合的方式有基于网络和基于主机的混合或者异常检测和误用检测的混合。

不管是哪种类型的 IDS,其工作过程大体是相同的,可以分为三个主要的环节,即信息收集、分类检测和决策,其中分类检测和决策环节是 IDS 的关键,都需要一定的人工智能技术来支持。

(1) 信息收集:入侵检测的第一步是信息收集,收集内容包括系统、网络、数据及用户活动的状态和行为。由放置在不同网段的传感器或不同主机的代理来收集信息,包括系统和网络日志文件、网络流量、非正常的目录和文件改变、非正常的程序执行。

(2) 分类检测:收集到的有关系统、网络、数据及用户活动的状态和行为等信息被送到检测引擎。检测引擎根据不同的检测机制进行检测,典型的方法有模式匹配、监督学习模型、半监督学习模型和离群点检测等。当然,在执行分类之前,需要在系统后台先进行模型训练,其可以离线完成。

(3) 决策:当检测到某种入侵行为时,控制台按照告警产生预先定义的响应措施,可以是重新配置路由器或防火墙、终止进程、切断连接、改变文件属性等,也可以是简单地发送告警。决策最主要的问题在于,检测器的召回率和准确率并不会达到 100%的效果,导致决策时可能产生不合适的措施。

3.1.3 数据集

一般认为数据质量决定了机器学习性能的上限，而机器学习模型和算法的优化最多只能逼近这个上限。因此在数据采集阶段需要对采集任务进行规划。在数据采集之前，主要是从数据可用性、采集成本、特征可计算性、存储成本的角度进行分析，以获得尽可能多的样本特征为基本目标。

入侵检测的数据采集方法取决于入侵检测系统的类型，即网络入侵检测和主机入侵检测系统。对于网络入侵检测，采用网络嗅探、网络数据包截获等方法获得流量数据。对于主机入侵检测，采用的方法比较灵活，既可以是操作系统的各种日志，也可以是某些应用系统的日志，还可以通过开发驻留于主机的应用软件等方法获得主机数据。因此，与网络连接、网络请求有关的特征，以及各类日志中的特征都是入侵检测常用的数据源。

这里介绍入侵检测领域常用的数据集，包括 NSL-KDD 等，这些公开的数据集为帮助研究人员比较不同的入侵检测方法提供了基准。NSL-KDD 数据集是通过网络数据包提取而成，由 M. Tavallaee 等于 2009 年构建，它克服了更早之前 KDD Cup 99 数据集中存在的一些问题。

NSL-KDD 共使用 41 个特征来描述每条流量，这些特征可以分为三组。

(1) 基本特征(basic features)，从 TCP/IP 连接中提取。

(2) 流量特征(traffic features)，与同一主机或同一服务相关。

(3) 内容特征(content features)，反映了数据包中的内容。

除此之外，每条流量都带有一个标签，即 normal 和 anomaly，表示相应的流量为正常或异常。该数据集中，异常流量又细分为 23 种攻击类别。

从特征工程的角度看，NSL-KDD 实际上已经完成了特征工程中的特征可用性、特征采集，以及衍生特征的定义和计算。使用该数据集进行检测实验，只要从特征清洗、特征选择或特征提取开始就可以。

NSL-KDD 每条流量的 41 个特征的含义如表 3-1 所示，表中列出了特征名称及其类型，其中 continuous 是连续数值型，symbolic 是符号类型。例如，protocol_type 属于 symbolic 类型，它的取值范围是 {'tcp','udp','icmp'}，是一种枚举值。

表 3-1 NSL-KDD 特征

特 征	类 型	特 征	类 型
duration	continuous	is_guest_login	symbolic
protocol_type	symbolic	count	continuous
service	symbolic	srv_count	continuous
flag	symbolic	serror_rate	continuous
src_bytes	continuous	srv_serror_rate	continuous
dst_bytes	continuous	rerror_rate	continuous
land	symbolic	srv_rerror_rate	continuous
wrong_fragment	continuous	same_srv_rate	continuous
urgent	continuous	diff_srv_rate	continuous

续表

特　征	类　型	特　征	类　型
hot	continuous	srv_diff_host_rate	continuous
num_failed_logins	continuous	dst_host_count	continuous
logged_in	symbolic	dst_host_srv_count	continuous
num_compromised	continuous	dst_host_same_srv_rate	continuous
root_shell	continuous	dst_host_diff_srv_rate	continuous
su_attempted	continuous	dst_host_same_src_port_rate	continuous
num_root	continuous	dst_host_srv_diff_host_rate	continuous
num_file_creations	continuous	dst_host_serror_rate	continuous
num_shells	continuous	dst_host_srv_serror_rate	continuous
num_access_files	continuous	dst_host_rerror_rate	continuous
num_outbound_cmds	continuous	dst_host_srv_rerror_rate	continuous
is_host_login	symbolic		

从 https://www.unb.ca/cic/datasets/nsl.html 下载数据文件，该数据压缩文件中包含的文件说明如下。

KDDTrain ＋.TXT：是完整的 NSL-KDD 训练集，除了 41 个特征外，还包括数据包类型的标签和难度等级。其中，数据包类型有 normal，以及 back、buffer_overflow、guess_passwd、portsweep、rootkit、satan、smurf、teardrop、warezclient、warezmaster 等入侵类型。难度等级表示每条记录分类时判断的难易程度，是一个[0,21]范围内的整数，数值越大表示该记录越容易分类，0 是最不容易分类的。整个数据集共 125 973 条记录，难度等级小于 15 的记录占 2.94%，可以看出绝大部分记录的分类标签都是比较确切的。

KDDTrain ＋.ARFF：与 KDDTrain ＋.TXT 大致相同，只是每条记录不包含难度等级，同时数据包类型的标签被归类为 normal 和 anomaly 两种。该文件带有 41 个特征的属性名和类型描述，可以直接在 Weka 中使用。

KDDTrain ＋_20Percent.TXT：是 KDDTrain ＋.txt 文件的 20%子集，实际上是 KDDTrain ＋.txt 前 20%的记录。

KDDTrain ＋_20Percent.ARFF：是 KDDTrain ＋.arff 文件的 20%子集。

KDDTest ＋.TXT：是完整的 NSL-KDD 测试集，包括攻击类型的标签和 CSV 格式的难度等级。

KDDTest ＋.ARFF：是完整的 NSL-KDD 测试集，带有 ARFF 格式的二进制标签。

KDDTest-21.TXT：是 KDDTest ＋.txt 文件的子集，其中不包括难度级别为 21 的记录，即该数据集中共有 21 个难度等级。

KDDTest-21.ARFF：是 KDDTest ＋.arff 文件的子集，其中不包括难度级别为 21 的记录，该数据集共包含 21 个难度等级。

3.1.4　数据预处理

对于分类任务来说，由于原始数据可能存在异常、缺失值以及不同特征的取值范围差异大等问题，对机器学习会产生影响，因此，在进行机器学习模型训练之前，需要先对数据

进行预处理。数据预处理的主要过程包括数据清洗、去量纲、离散化等。

1. 数据清洗

对采集到的数据进行清洗，主要工作包括缺失值处理和异常值处理。

1）缺失值处理

缺失值是指样本中存在某个或某些特征没有值的情况，对此，可以采取的处理策略有删除数据、数据填充。

如果整个数据集中的某个特征值缺失得较多，就可以简单将该特征舍弃。如果包含缺失值的记录不多，则可以采用一些常用的填充策略。典型的方法有固定值填充、均值填充、中位数填充、上下数据填充、插值法填充和随机数填充等。这些方法的基本出发点是利用该特征在整个数据集中的统计量来填充，例如中位数就是把非缺失的特征值进行排序后取中间位置上的数作为缺失记录的特征值。

2）异常值处理

异常值是指样本中的某个特征取值与其他样本有显著差异，例如某个记录的年龄字段为 200 岁，某城市的气温为 100℃等。

针对这种情况可以采取的策略有按照缺失值处理、采用其他样本的平均值或最大值等统计量来代替，也是一些启发式的处理方式。

2. 去量纲

数据集中不同属性的取值范围可能存在很大的差异，例如用米为单位度量的身高和以千米度量的两个城市之间的距离。这种差异会导致机器学习模型的目标函数在某些维度上取值范围远远大于其他维度，当进行梯度下降时，收敛慢，训练时间过长。

去量纲的要求是使不同取值范围的特征值转换到同一规格，一般是[0,1]或[−1,1]等。常见的去量纲方法有归一化和标准化。

通过归一化把原始数据转换为单位向量，主要有最大最小缩放、对数变换、反正切变换，计算公式分别如下。

$$x' = \frac{x - \mathrm{Min}}{\mathrm{Max} - \mathrm{Min}} \tag{3-1}$$

$$x' = \log x \tag{3-2}$$

$$x' = \frac{2}{\pi}\arctan x \tag{3-3}$$

最大最小缩放用于线性数据，对数变换和反正切变换用于非线性数据。

当原始数据服从正态分布时，还可以使用标准化去量纲，首先计算原始数据的均值 μ 和标准差 S，然后使用式(3-4)对数据进行标准化，即转换成标准正态分布。

$$x' = \frac{x - \mu}{S} \tag{3-4}$$

3. 离散化

当我们使用某些机器学习模型进行训练时，要求相应的训练数据必须为离散型数据，

例如决策树、朴素贝叶斯等算法都基于离散型数据。

离散化方法有等宽法、等频法和基于聚类的方法等。

等宽法,顾名思义就是将特征值从最小值到最大值按次序分成具有相同宽度的 n 个区间。例如[0,59]按 3 等分被划分为[0,19]、[20,39]、[40,59]。等频法根据数据的频率分布进行排序,然后按照相同频率进行区间划分,因此能保证每个区间的样本数量相同。

基于聚类的方法也可以将连续属性值转换为离散值。通过聚类算法及聚类有效性指标(validity index)进行最佳簇的划分,把同一个簇内的样本按同一个值来处理,即簇的标识或聚类中心。

4. 哑变量

哑变量(dummy variables) 也称虚设变量,通常取值为 0 或 1。例如,反映性别的哑变量可以取值为 0:男性,1:女性。

在机器学习中,经常会遇到类别型特征,如入侵检测数据集中的网络协议(protocol_type),它的取值为{'tcp','udp', 'icmp'},这种字段不能直接输入给分类器。转换方式就是增加哑变量,并进行 one-hot 编码。对于具有三种取值的 protocol_type 字段,可以拓展为三个字段,并编码。如表 3-2 所示,表中的三行分别为 tcp、udp 和 icmp 的编码。

表 3-2 protocol_type 的哑变量编码

protocol_type_tcp	protocol_type_udp	protocol_type_icmp
1	0	0
0	1	0
0	0	1

3.1.5 特征工程

样本特征数量的多少显然对机器学习模型性能会产生一定的影响。当特征数量太少时,样本在较小的特征空间内可能重叠在一起。如图 3-1 所示,在二维空间线性可分的两类样本,当缩减到一维时,变得线性不可分,最终导致分类器都失效;反之,当特征数量太多时,属于同类样本的数据在特征空间中变得稀疏,导致类别边界模糊,分类性能受到影响。此外,特征数量多,特征之间存在相关性的可能性增加,模型的复杂度也会变大。

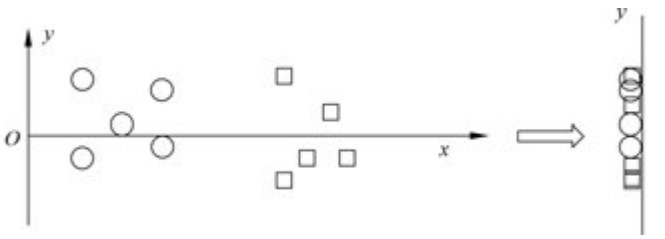

图 3-1 减少特征维数导致样本重叠

针对网络入侵检测应用，其特征数据通常来自多个不同的软硬件设备、不同的应用系统、不同的日志系统，但是都与攻击行为有一定联系，由此可能导致特征之间存在一定的相关性。例如，针对某个端口的大量并发连接请求，也必然引起内存使用量的增加。又如，Web 服务器通常使用默认端口 80 进行监听，不同服务器一般有默认端口，由此服务器类型和端口就存在一定的相关性。因此，构造合适的特征空间也是很有必要的。

特征选择和特征抽取是特征工程的两个重要的方面，目的都是寻找合适的样本表示空间。它们的最大区别是是否生成新的属性。特征提取通过变换的方法获得了新的特征空间，如 PCA、NMF 等。特征选择只是从原始特征集中选择出部分子集，没有生成新的特征，主要有筛选（filter）式、包裹（wrapper）式和嵌入（embedded）式。信息增益属于一种筛选式选择方法。具体的计算方法在很多机器学习方面的书中都有介绍，这里不再赘述。

3.1.6　AI 模型及其应用开发

以 NSL-KDD 数据集为例，选择其中的部分数据构造了二分类问题、多分类问题。在二分类中，训练集共有 125 973 条记录，类别是正常和异常两类。经过去除缺失值记录、归一化等数据处理后，进行特征选择，然后以 SVM、决策树、逻辑回归和随机森林作为分类器进行了训练。对另外的测试数据文件进行测试。

```
def Entropy(X):
    p_ = X.value_counts() / X.shape[0]
    return sum((-1) * p_ * np.log2(p_))

#信息增益率
def Gain(data, str1, label):
    E = data.groupby(str1).apply(lambda x: Entropy(x[label]))
    p_ = data[str1].value_counts() / len(data[str1])
    print(p_)
    e_ = sum(p_ * E)
    print(e_, Entropy(data[label]))
    return Entropy(data[label]) - e_
```

执行特征选择：包含信息增益和方差阈值选择法，后者比较简单，是 sklearn. feature_selection 中提供的函数。

如果使用信息增益，可以直接在 train_data 上进行，如下：

```
    FeatureSet = []
    for feature in range(0,41):
        FeatureSet.append(feature, Gain(train_data, feature, 41))
    Sort(FeatureSet)
    #得到按信息增益值排序的特征序列，可取前 K 个，K 自定

def Feature_select(X_train, X_test):
    #去除出现次数的方差小于指定值的特征
```

```
    selector = VarianceThreshold(threshold = 0.000001)
    selector.fit(X_train)
    print(selector.variances_)
    X_train = selector.transform(X_train)
    X_test = selector.transform(X_test)
    print(X_train.shape)

if __name__ == "__main__":
    '''
    1.数据处理部分
    '''
    warnings.filterwarnings('ignore')
    train_data = pd.read_csv("train - binary.txt", header = None)
    test_data = pd.read_csv("test - binary.txt", header = None)
    train_data.dropna(inplace = True) # 删除有缺失值的行
    test_data.dropna(inplace = True)

    '''
    2. 对原始数据集特征进行归一化,特征选取,产生训练集、测试集
      一种更优的特征子集是 features = [0,2,3,4,10,11,13,16,26,29]
    '''
    features = [i for i in range(41)]
    x_train = pd.DataFrame(train_data, columns = features)
    y_train = train_data[41]
    x_test = pd.DataFrame(test_data, columns = features)
    y_test = test_data[41]

    scaler = Normalizer().fit(x_train)
    x_train = scaler.transform(x_train)
    scaler = Normalizer().fit(x_test)
    x_test = scaler.transform(x_test)

    Feature_select(x_train, x_test)
    '''
    3.构建分类器,测试性能.这里演示了 SVM、决策树、逻辑回归、随机森林,
      可以自行实现神经网络模型等更多分类器并进行测试
    '''
    svm = SVC()
    svm.fit(x_train[:1000], y_train[:1000])
    print('F1 (SVM): %.4f' % f1_score(y_test, svm.predict(x_test)))

    dt = tree.DecisionTreeClassifier()
    dt.fit(x_train, y_train)
    print('F1 (Decision Tree): %.4f' % f1_score(y_test, dt.predict(x_test)))

    lr = LogisticRegression()
    lr.fit(x_train, y_train)
    print('F1 (Logistic Regression): %.4f' % f1_score(y_test, lr.predict(x_test)))

    rf = RandomForestClassifier(n_estimators = 47)
    rf.fit(x_train, y_train)
    print('F1 (Random Forest): %.4f' % f1_score(y_test, rf.predict(x_test)))
```

在 sklearn 框架内，上述程序可以很容易拓展到多分类情况，需要改动的代码是计算 F1 值的函数 f1_score。该函数原型如下：

```
sklearn.metrics.f1_score(y_true, y_pred, labels = None, pos_label = 1, average = 'binary',
sample_weight = None)
```

对于二分类问题，average 设置为 binary；对于多分类问题，则需要设置为 macro 或 micro 来计算宏观或微观平均。如果需要考虑类别的不平衡性，按照类别的加权平均，则使用 weighted。

3.1.7 入侵检测问题与对抗演进

尽管机器学习方法实现了对入侵行为和正常访问的分类识别，但是仍存在一些机器学习难以解决的问题，概述如下。

(1) 误报率高、漏报率高。各种机器学习模型仍存在较高的误报率和漏报率，并且对于参数敏感。特别是对于未知的入侵行为的感知能力弱，已成为制约入侵检测发展的关键技术问题。

(2) 自学习能力差。添加 IDS 检测规则或特征常依赖于手工方式且更新缓慢，限制了 IDS 的可用性。

(3) 从检测到决策的困难。入侵检测的最终目标是为安全防御提供支持，而检测技术中的误报率和漏报率高的问题，使得自动化决策可能影响正常数据的流动，也可能导致未能及时阻断入侵行为。

(4) 自身易受攻击。IDS 本身是存在漏洞的软件程序，它容易成为黑客攻击的目标，一旦黑客攻击成功，它所管理的网络安全就不能得到保证。

因此，机器学习、人工智能方法在解决此类实际问题时仍有很多需要深入研究的技术。同时，也是因为这些问题的存在，使得针对 AI 模型的攻击成为入侵检测应用中的新对抗场景。

3.2 SQL 注入攻击与检测

3.2.1 概述

360 互联网安全中心在《2025 年度网络安全漏洞分析报告》中指出，SQL 注入攻击占 Web 应用安全问题的 13.42%。根据国家信息安全漏洞共享平台，共有 24 102 条漏洞与 SQL 注入相关，占 Web 应用漏洞的 58.97%。在这些漏洞中，仍有 19 094 个漏洞未被修复。由此可见，SQL 注入仍是网络安全中的重要问题。

SQL 注入是指攻击者利用 Web 网页对于输入数据处理时存在的漏洞，向数据库发起恶意请求。这种漏洞一般是对输入数据没有进行过滤处理或者处理规则不完善，将攻击者输入的恶意 SQL 语句或参数注入查询命令中并传给 Web 服务器，Web 程序执行被注入的 SQL 语句。当注入的 SQL 带有恶意性时，在数据库端的执行最终导致信息泄露

或数据库系统被破坏。

由于 SQL 数据库在 Web 应用中的普遍性,使得 SQL 攻击在很多网站上都可以进行。并且这种攻击技术的难度不高,但攻击变换手段众多,危害性大,使得它成为网络安全中比较棘手的问题。

3.2.2 SQL 注入方法

结构化查询语言(Structured Query Language,SQL)是一种用于与数据库进行数据交互的语言,而 SQL 注入就是指利用数据库之外的其他外部接口,将 SQL 要素绑定到接口并传入数据,使得接口程序构建并发起带有注入信息的 SQL 请求,从而达到入侵数据库的目的。

Web 脚本是 SQL 注入攻击中常见的一种外部接口,在这种情况下,SQL 注入攻击者通过 Web 页面输入一些特定的字符,当 Web 服务器没有对此输入进行合法性检验时,它们就形成特定的 SQL 语句。最终 SQL 语句被发送到数据库引擎并执行,从而产生不符合预期的数据库操作。

图 3-2 实现 SQL 注入的 Web 页面

例如,某个登录页面包含了用户名和密码的输入框,如图 3-2 所示。

假如对登录用户身份进行合法性验证的 SQL 语句为

```
select * from user where name = '{ $ name} ' and password = { $ password}
```

其中,name 和 password 分别为字符串和数字型,对应于该页面的两个输入框内容。

攻击者即使没有该网站的用户账号和密码,也可能绕过账号验证而获得相应登录权限。只需在登录提交表单中,将用户名输入为一个随意的字符串,如 asndfas,密码设为 1 or 2=2。此登录验证的 SQL 语句就被构造为

```
select * from user where name = 'asndfas' and password = 1 or 2 = 2
```

在这个 SQL 查询中,由于 2=2 的条件恒成立,因此,SQL 执行的结果是返回 user 表中的所有记录。在 Web 脚本的后续处理中如果认为有记录返回而允许登录,那么这样的输入攻击就可以绕开验证而获得合法的登录权限,而且攻击者不需要知道真正的用户账号或密码。

除了可以注入参数到 SQL 中,在一定条件下,也可以注入 SQL 语句来进行数据库结构的猜测。例如攻击者想知道数据库中是否存在指定的表或表的字段名,在确定数据表中存在用户名和密码为 bbb/2345 的记录情况下,攻击者可以通过如下的密码注入,来检测 users 表中是否存在 emails 字段。

```
select * from users where username = 'bbb' and password = 2345 and exists(select emails from
users)
```

由此可以进一步看出,攻击者可以注入恶意数据库操作来实现更严重的效果,如执行一些数据库操作,导致数据丢失。例如在密码框中输入 2345; drop table tmp,从而形成了如下的注入 SQL,其中注入的分号是将 SQL 指令分成多条指令执行。

```
select * from users where username = 'bbb' and password = 2345; drop table tmp
```

可以看出,当 Web 连接数据库的用户具有数据库执行 drop table 的权限时,这条语句中的 drop table tmp 将会被执行,从而实现删除 tmp 表的操作。用此类方法可以对数据表内容或者结构进行删除,也可以使用 Update 等语句对数据表信息进行修改。

3.2.3 SQL 注入检测与 AI 安全起点

SQL 注入的检测通常要对输入的内容进行校验,其中较为有效的是对请求数据格式或者内容进行规则处理。目前主要的检测方法如下。

1. 针对特定类型的检查

考虑到 SQL 注入是在特定的 Web 页面输入框中实现的,每个输入有其特定的格式要求。因此,可以对页面变量的数据类型、数据长度、取值格式、取值范围等进行检查。例如 where id ={$ id},对于输入的 id 进行类型检查。只有当这些要求都通过检查之后,才把请求发送到数据库执行。这种方法能对有特定的数据格式的输入起到防 SQL 注入的作用。但其局限性较大,对每个网页程序接口输入都进行格式判断,工作量较大,并且存在较大的遗漏或不准确性。

2. 对特定格式的检查

对于格式有明确要求的输入,如邮箱或者电话等,可以采用正则表达式过滤方法,排除不符合要求的变量。正则表达式过滤的方法也可以过滤掉一些常见的注入,例如对于 ' or 1=1 之类的注入,其匹配的正则表达式为('\s+)? or\s +[[:alnum:]]+ \s *= \s *[[:alnum:]]+\s *(--)? ,只要拒绝符合该正则表达式的输入即可达到防止 SQL 注入的目的。这种方式的优点是可以过滤已知的各种注入方法,但是不能过滤未知的注入。缺点是这种方法也会将带有符合过滤正则表达式的合法输入过滤掉,例如用户的博客中的某一句子带有 ' or 1=1 --,那么该句子会被错误地过滤掉。

3. SQL 预编译的防御方法

SQL 预编译的基本思想是创建 SQL 语句模版,将参数值用“?”代替,例如“select from table where id = ?”,然后经过语法树分析、查询计划生成,缓存至数据库。这种方法不论输入内容包含什么,总是被当作字符串。这样,用户传进来的参数只能被视为字符串用于查询而不会被嵌入 SQL 中再去执行语法分析。一些 Web 框架如 Hibernate、

MyBatis 等已经实现了参数化查询,是目前比较有效的防止 SQL 注入的方法。但是考虑到程序员的编程习惯、预编译对资源的占用以及所选择的框架等因素,仍存在需要采取字符串拼接生成 SQL 的场景。

4. 机器学习方法

机器学习方法把 SQL 注入检测看作一个二分类问题,从而按照机器学习的一般流程进行设计。主要环节包含训练数据收集与标注、特征提取、分类器选择与训练以及执行分类等,具体过程将在 3.2.4 节展开说明。由于 SQL 注入攻击的危害性大,注入模式变化多端,作为防御者,必然要采取足够的智能手段才能应对这些注入攻击。这同时也开启了人工智能对抗的新阶段,是人工智能安全的起点。

3.2.4 SQL 语句的特征提取

使用机器学习进行 SQL 注入的检测,首先需要解决 SQL 语句特征表示的问题。主要的方法有基于图论的方法、基于文本分析的方法等。

1. 基于图论的方法

在文本分析、关键词提取、社交网络分析等应用中,利用图来表示其中的特征是很常见的。SQL 语句具有文本特征,因此有文献提出基于图论的 SQL 语句特征提取方法。该方法把 SQL 查询建模成标记图,进而生成以标记为节点、节点间的交互为带权边的图,利用该图实现 SQL 语句的转换和表示[1]。

首先定义 SQL 语句中的标记(token),把 SQL 中的关键字、标识符、操作符、分隔符、变量以及其他符号都称为标记。这样,一条 SQL 查询,无论是真正的查询还是注入的查询,都是一个标记序列。检测的基本思路就是,对这些真正查询和注入查询的序列进行特征提取,然后在特征空间中构建识别注入查询的分类器模型。

所定义的部分 token 及其规范化的符号如表 3-3 所示,包含了用户定义的对象、SQL 关键字、字符类型、运算符和符号等。

表 3-3 SQL 语句符号替换对照表

符　　号	替　换　为	符　　号	替　换　为
整数	INT	SQL 关键词、函数	转换为大写
IP 地址	IPADDR	<	LT
十六进制数	HEX	>	GT
系统表	SYSTBL	( )	去除
用户表名	USRTBL	,	CMMA
用户表中的字段名	USRCOL		

对于一条 SQL 语句,使用 token 对照表进行转换,同时对语句做特殊处理。

(1) 对能够匹配的括号对进行删除;对不能匹配的括号给予保留,并转化成标记。

(2) 空注释(包含只有空白字符的注释)可以被删除,但非空注释必须保留。这是因

为攻击者通常在注入代码中嵌入空注释来做混淆(例如/ ** / OR / ** /1/ ** /=/ ** /1)，企图绕过检测。

以下是两个替换的例子。

(1) select * from books where price > 20.5 and discount < 0.8 规范化为

```
SELECT STAR FROM USRTBL WHERE USRCOL GT DEC AND USRCOL LT DEC
```

(2) select count(*),sum(amount) from orders order by sum(amount)规范化为

```
SELECT COUNT STAR CMMA SUM USRCOL FROM USRTBL ORDER BY SUM USRCOL
```

最终，构建了 686 个不同的标记。每个标记都被看作最终数据集中的一个属性(维度)。

定义(标记图)：标记图是一个有权图 $G=(V,E,w)$，其中 V 中的每个顶点对应一个规范化序列中的标记，E 是边的集合，并且 w 是一个定义边权重的函数。如果 t_i 和 t_j 在一个长度为 s 个标记的滑动窗口中同时出现，则称在标记 t_i 和 t_j 间有一条权重为 w_{ij} 的边。如果在窗口滑动过程中，t_i 和 t_j 间已经有一条边了，则它的权重要加上新的权重。

定义(无向图)：在无向标记图中，如果标记 t_i 和 t_j 中有一条边，则它具有对称的权重 $w_{ij}=w_{ji}$。如果在同一个长度为 s 个标记的滑动窗口中出现了 t_i 和 t_j，则不进行权重累加。

对于有向图的构建，按照 SQL 语句，从左到右寻找标记并建立有向边。

定义(有向图)：在一个有向标记图中，当一个长度为 s 个标记的窗口滑过时，如果 t_i 出现在 t_j 之前，则认为 $t_i \rightarrow t_j$ 形成一条权重为 w_{ij} 的边。边 $t_i \rightarrow t_j$ 和 $t_j \rightarrow t_i$ 的权重是独立地计算的。

计算含权图中两个节点间连接权重的两种加权方法如下：①在 s 滑动窗口内，每个标签具有相同的权重；②在 s 滑动窗口内，离得近的标签权重大，离得远的标签权重小。以两个 SQL 标记序列为例，第一个标志在滑动窗口内与后续标记的连接关系及权重，如图 3-3 所示，(a)采用的是均匀权重，(b)采用的是按比例权重。

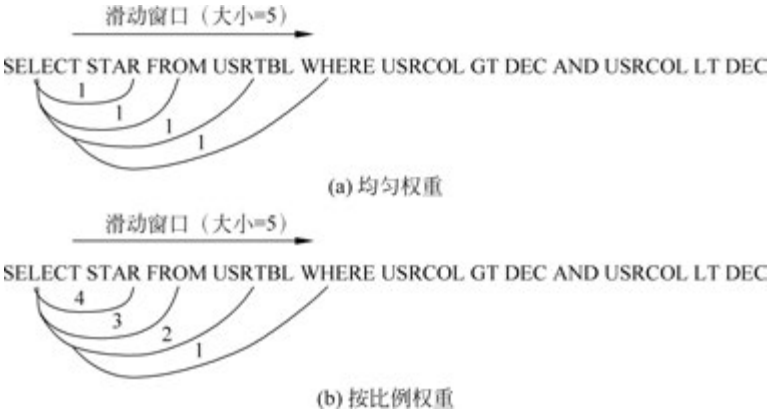

图 3-3　均匀权重与按比例权重

在标记图构造完成之后,所生成的图的示例如图 3-4 所示,其中图 3-4(a)、图 3-4(b)分别是正常查询和注入查询的标记图。将每个节点按照一定的特征进行表示,相应的特征值应当反映节点的重要性。可以选择的特征量包括度数、介数、紧密度等中心性度量,这些也经常用于衡量文本、社交关系中的重要性。但对于 SQL 注入而言,考虑到在 SQL 数据库上执行而造成对服务器的资源消耗,因此,可以选择计算量小的度数。

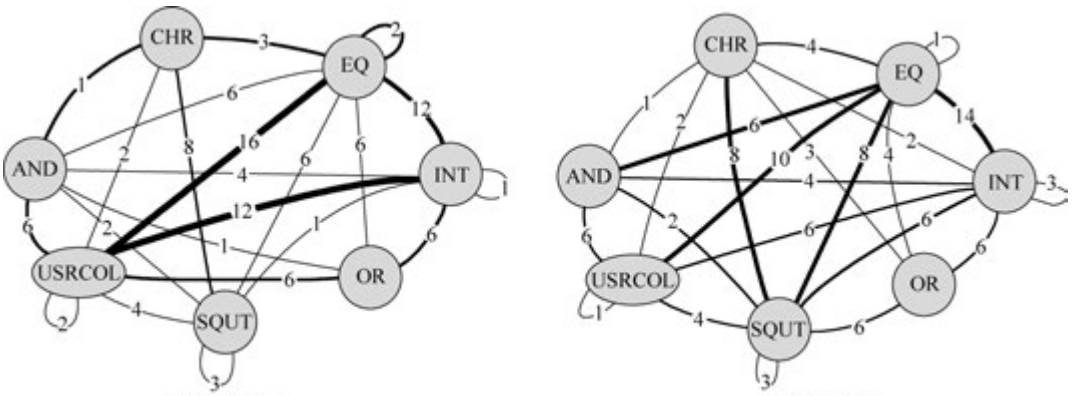

图 3-4 正常查询和注入查询的标记图[1]

2. 基于文本分析的方法

由于注入内容是一种文本信息,其语法基本遵循 SQL 语言,而非杂乱无章的内容。从这点来看,它与自然语言类似。因此,可以尝试按照自然语言文本分类的方式来进行 SQL 注入的检测。

把 SQL 语句注入的请求信息进行分割,按照逻辑顺序进行切分,在逻辑上存在间隔的地方加上空格。

例如对于以下 Web 日志:

```
-- post - data
"Login = 'and'1' = '1～～～&Password = 'and'1' = '1～～～&ret_page = 'and'1' = '1～～～&qu
erystring = 'and'1' = '1～～～&FormAction = login&FormName = Login"
```

转化为

```
- post - data "Login = ' and '1' = '1～～～&Password = ' and '1' = '1～～～&ret_page = ' and '1' =
'1～～～&querystring = ' and '1' = '1～～～&FormAction = login&FormName = Login
```

把这些从 Web 日志中提取出来的字符串按照标点符号进行切分,最终获得其中的词汇特征集。这样的做法是把这些字符串当作文本来处理。

接下来,采用普通的文本分类技术,使用信息增益、方差阈值等特征选择方法选择有利于分类的 Top k 个特征,从而完成文本向量空间的构建。

从文本的角度,当然也可以利用文本分类中的经典神经网络来进行 SQL 注入的检

测。例如,把 SQL 语句当作文本,使用 TextCNN 进行分类。如前所述,TextCNN 的输入文本信息可以是标记化之后的 SQL 语句、经过空格分隔之后的 SQL 语句或经过空格和逻辑运算符分割之后的 SQL 语句。

3.2.5 AI 模型及其应用开发

数据集来自一个网站收集的链接请求,只有 normal/attack 两类,分别对应标签 0/1。该数据集共 480 条记录,有注入记录 339 条和正常记录 141 条。以下两条记录分别是注入和非注入样本。

```
-- post - data ""username = test' % 20or % 201 = 1; ～～～&password = '''～～～"" http://
endeavor.cc.gt.atl.ga.us:8080/checkers_current/servlet/processlogin

-- post - data ""username = and&password = test"" http://endeavor.cc.gt.atl.ga.us:8080/
checkers_current/servlet/processlogin
```

基本的任务是构建分类器来完成 SQL 注入语句的检测,区分正常访问(normal)和含 SQL 注入攻击(attack)的网络请求。

基本思路是,把整个训练文本集进行切分,转换为 tf-idf 向量,然后使用各种分类器进行训练和测试。

(1) 数据处理部分。

```
train_data = pd.read_csv("train.txt", header = None, sep = ",")
test_data = pd.read_csv("test.txt", header = None, sep = ",")
train_data.dropna(inplace = True) #删除有缺失值的行
test_data.dropna(inplace = True)
```

(2) 文本-向量转换处理,使用 sklearn 提供的 TfidfVectorizer 完成向量表示。可以自行实现 Word2Vec 等更多方法。

```
x_train = list(train_data[0])
y_train = train_data[1]
x_test = list(test_data[0])
y_test = test_data[1]
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(x_train + x_test)
point = len(x_train)
x_train = X[:point]
x_test = X[point:]
#查看特征空间
print("特征空间: ")
print(vectorizer.get_feature_names())
```

(3) 构建分类器,测试性能。可以利用神经网络模型等更多分类器进行测试,由于数据质量较高,SVM 分类器和决策树的 F1 值分别可以达到 0.9406 和 0.9914。

```
svm = SVC()
svm.fit(x_train, y_train)
print('F1 (svm): %.4f' % f1_score(y_test, svm.predict(x_test)))

dt = tree.DecisionTreeClassifier()
dt.fit(x_train, y_train)
print('F1 (Decision Tree): %.4f' % f1_score(y_test, dt.predict(x_test)))
```

特征空间的部分维度如下：

```
'20delete_priv', '20drop_priv', '20exec', '20fieldname', '20file_priv', '20from', '20grant_
priv', '20having', '20host', '20index_priv', '20information_schema', '20inner_join',
'20insert', '20insert_priv', '20into', '20join', '20left', '20limit', '20master', '20mysql',
'20or', '20outer', '20password', '20process_priv', '20references_priv', '20reload_priv',
'20select', '20select_priv', '20set', '20show', '20shutdown_priv', '20string', '20tab',
'20table', '20table_name', '20tablename', '20tables', '20top', '20union',
```

可以看出，这里选择出的特征都是通常用于 SQL 语句的词汇，因此用来判断 SQL 注入是比较合适的。

3.3 虚假信息检测

3.3.1 概述

虚假新闻、谣言等不实信息在互联网上层出不穷，虚假信息的检测不仅具有明显的应用需求，也是人工智能技术非常好的试验场。因此，近年来虚假信息检测方法得到了广泛关注。

不同于网络入侵等网络安全问题，虚假信息类安全问题并非数据层面的安全问题，而是在数据之上，属于内容安全范畴。内容安全和行为安全有时并不是完全分开的，例如考虑到谣言内容识别时，其传播行为会表现出一定的特征，因此在谣言检测中也可以使用谣言传播行为的特征，内容安全和行为安全混杂在一起。可以进一步从谣言传播、水军、特定群体的形成与演变等行为安全角度来提升虚假信息检测效果。

从人工智能技术角度看，内容安全主要是基于文本处理技术。从文本中提取关键词、命名实体、主题特征等，使用各种文本表示模型给出数学表示，并最终选择合适的分类器进行训练和分类。在特征方面，通常可以根据文本的不同部分，例如标题、段落和结尾部分，分别进行特征处理。

虚假新闻检测与虚假信息、谣言信息检测有一定的相似性，但虚假信息包含较多方面，包括虚假新闻、虚假评论等，谣言信息可能是虚假信息，也可能是真实信息。从技术手段上看，这些检测方法有一定相似性。最基本的方法是仅利用信息内容，主要是文本内容，从文本分类的角度进行检测。

更进一步，需要针对不同类型信息，引入更多特征。对于谣言而言，典型的特征还有信息传播特征，如传播的速度、涉及的人群、传播中的重要人物等。对于虚假评论而言，其

他可用的特征包括评论人的行为，如评论中的语气、评论数量、涉及的商品等。对于虚假新闻，可以考虑长文本所具有的特征，例如篇章、主题和文本中特定实体信息等。

3.3.3 节和 3.3.4 节提供了两个例子，分别是基于统计学习的检测和基于多任务学习的检测，并在天池 AI 学习平台的课程案例中给出了完整的代码和数据集，具体的访问和使用方法见第 18 章的说明。

3.3.2 虚假信息与 AI 安全

虚假信息虽然不同于前述两类网络攻击，但是从本质上看，编制虚假信息也有一定的动机，并且这种动机与虚假信息编制者(攻击者)的利益有密切关系。例如，攻击者采取各种“吸引眼球”策略编制虚假信息，促进不实信息传播和扩散，从而达到攻击者所预期的目的。

在过去很长时间里，虚假信息大都依赖于人工编制。编者会充分考虑大众的接受度、同情心等多方面的因素。由于虚假信息的多样性和复杂性，互联网内容管理机构必然需要采用人工智能技术进行虚假信息检测。因此，从人工智能安全观的角度来看，人工编制虚假信息与虚假信息检测是一种 AI 安全的初始对抗场景。

3.3.3 基于统计学习的检测

1. 数据集

本节所使用的数据集包含了 2096 条来自 68 个不同网站的新闻信息，新闻发布日期是 2016 年 10 月 26 日—11 月 25 日。每条新闻经过了标注，共有 801 条真实新闻，1294 条虚假新闻，另有一条新闻未作标记。标签集为{'Fake','Real',nan}。每条新闻有 12 个属性，属性特征及字段名称如表 3-4 所示。

表 3-4 数据集中新闻属性及字段名称

属 性	字 段 名 称	取 值 说 明
作者	author	
发布日期	published	
标题	title	
文本	text	
语言	language	{nan,'spanish','french','ignore','german','english'}
来源	site_url	
主要图像的 URL	main_img_url	
类型	type	conspiracy、hate、satire、junksci、state 等表示新闻类型
去除停用词后的标题	title_without_stopwords	
去除停用词后的正文	text_without_stopwords	
是否含有图片	hasImage	0(没有图片)、1(有图片)
标签	Label	标注结果

2. 数据特征分析

对于给定的数据集,可以先利用可视化技术对数据集进行探索性分析,以便对各个属性特征有深入了解,有助于设计更好的特征工程。

首先读入文件:news_articles.csv,删除表格中含有 nan 的记录,共得到 2045 条记录。

利用下面的语句,可以使用饼状图查看不同类型新闻的分布情况,如图 3-5 所示。可以看出,bs、conspiracy、bias 和 hate 所占比例超过了 75%。

```
df['type'].value_counts().plot.pie(figsize = (8,8), startangle = 75)
```

利用 WordCloud 组件,生成词云图,用词云查看新闻中的关键词分布,如图 3-6 所示。

```
wc = WordCloud(background_color = "black", max_words = 100,
            max_font_size = 256,random_state = 42, width = 2000, height = 2000)
wc.generate(' '.join(df['text_without_stopwords']))
```

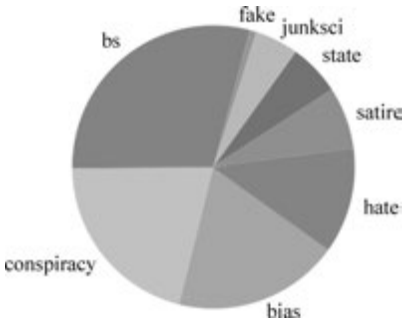

图 3-5 新闻类型的分布

图 3-6 词云图

3. 数据处理与分类

对清洗后的数据集进行训练集与测试集的划分,为了简化示例,这里选取了 url 和文本特征,将两列合成一列作为新的特征 source,并用 tf-idf 词向量处理数据集,将向量数据存在 DataFrame 中,便于后续训练和测试。

```
#分割训练集和测试集,并用 tf-idf 向量表示,保存到 tfidf_df 中
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.30)
tfidf_vect = TfidfVectorizer(stop_words = 'english')
tfidf_train = tfidf_vect.fit_transform(x_train)
tfidf_test = tfidf_vect.transform(x_test)
tfidf_df = pd.DataFrame(tfidf_train.A, columns = tfidf_vect.get_feature_names())
```

为了展示分类器的应用，这里以 SVM、AdaBoost、RandomForest、XGBoost 为例，可以进一步查看不同分类器对结果的影响。

```
    # AdaBoost
    Adab = AdaBoostClassifier(DecisionTreeClassifier(max_depth = 10), n_estimators = 5,
random_state = 1)
Adab.fit(tfidf_train, y_train)
y_pred2 = Adab.predict(tfidf_test)
ABscore = metrics.accuracy_score(y_test, y_pred2)
print("accuracy: %0.3f" % ABscore)
    # RandomForest
    Rando = RandomForestClassifier(n_estimators = 100, random_state = 0)
    Rando.fit(tfidf_train, y_train)
    y_pred3 = Rando.predict(tfidf_test)

    #XGBoost
    xgb_clf = XGBClassifier()
    xgb_clf.fit(tfidf_train, y_train)
    y_pred4 = xgb_clf.predict(tfidf_test)
```

根据测试结果可以看出，在这几种不同的分类器中，AdaBoost 可以获得最好的分类性能，准确率达到 96.9%；而随机森林和 XGBoost 相当，都获得 85.3%的准确率。具体的实现方法是调用 sklearn 中的功能，在使用之前先加载如下包，其中 RandomForestClassifier 和 AdaBoostClassifier 在集成学习中。

```
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from xgboost import XGBClassifier
```

3.3.4　基于多任务学习的检测

小样本是网络安全检测中的普遍问题，本节介绍小样本多任务学习在虚假新闻中的运用方法。

1. 数据集

2017 年，William Yang Wang 公布了一份较大的数据集 LIAR，其中共包含了 12 836 条新闻[2]。该数据集是从一个事实核查网站 PolitiFact 收集的，包括简短陈述，例如新闻稿、电视或电台采访、竞选演讲等，并包含元数据。

除了新闻的文本内容外，LIAR 数据集还提供了丰富的上下文信息，如作者、党派、作者历史信用表现等，每条新闻均由专业的新闻工作者审核，并赋予一个反映新闻真实程度的标签，按照从假到真分为以下六个等级：pants-fire、false、barely-true、half-true、mostly-true 和 true。此外，每条新闻均提供了详细的鉴定报告，阐述了新闻产生的背景以及新闻中相关论点的背景知识，是研究虚假新闻检测较为可靠的数据集。该数据集字

段及例子的解释如表 3-5 所示。

表 3-5 LIAR 数据集字段

字 段	取 值 样 例
所在文件	8616.json
真实程度	mostly-true
文本内容	The economy bled $24 billion due to the government shutdown
主题	economy、federal-budget、health-care
来源	Doonesbury
作者	
党派	Democrats、Republicans 或无
工作	
所在州	
历史信用	0 0 2 4 0 (在 pants-fire、false、barely-true、half-true、mostly-true 信息的计数)
上下文	a Doonesbury strip in the Sunday comics

本节基于此数据集来验证模型的有效性。LIAR 数据集中共有三份数据集,分别为训练集、验证集和测试集。使用训练集来训练多任务学习模型,验证集用来验证训练模型的效果并选择最优参数,实验中的相关性能指标均为模型在测试集上计算得到。

2. 深度神经网络模型设计

多任务学习可以利用多个学习任务中的共享特征信息来提升相关任务的泛化性能,从而进一步提高模型的整体性能。在虚假新闻信息样本有限的情况下,运用多任务学习是值得尝试的做法。

1) 源任务的选择

基于多任务的思路,必须为虚假新闻检测寻找其他相关或相似的源任务,然后才能进行任务之间的参数共享和模型训练。以同一个数据集为基础进行源任务设计,应当考虑到源任务所学习到的特征有利于提升目标任务(即虚假新闻检测)的准确性,这种提升是相对于目标任务仅利用与虚假新闻直接相关的特征而言的。

由于不同主题新闻出现虚假信息的可能性差别较大,例如娱乐类新闻出现虚假信息的可能性比科技类新闻要大得多,社会突发事件出现虚假信息的可能性比其他类型主题新闻要大得多。因此,如果能够把主题特征提取出来,并与虚假特征进行融合,将有可能提升虚假新闻的检测效果。由于新闻的主题特征并不直接存在于新闻文本中,提取新闻主题特征本身就是一个机器学习任务。

此外,多任务学习的目的是利用源任务的充足数据来挖掘更多特征,以利于向目标任务共享有效参数。尽管在本问题中,两个任务使用同一个数据集,但是源任务可以从数据集中进一步构造出它的训练数据。

综上所述,在虚假新闻分类的多任务学习设计中,可以把新闻主题分类作为源任务,把虚假新闻分类作为目标任务。基于新闻真实性与新闻主题之间的关系,运用深度学习

技术，构建多任务学习模型，使模型可以同时预测新闻的真实性以及新闻的主题。

2）模型结构

由于深度学习模型能够学习不同层面的特征，具有天然的参数共享机制，因此在多任务学习模型中，通常以深度神经网络作为基本模型。在多个任务之间共享底层的隐藏层，并且针对不同任务设计相应的神经网络来处理高层特征。一般来说，所选择的源任务应当有较充足的数据，在共性特征共享时，先由源任务训练参数，然后复制给目标任务。目标任务只更新模型中与任务相关的层，而源任务可以同时更新与共享和任务相关的层。

模型结构如图 3-7 所示。

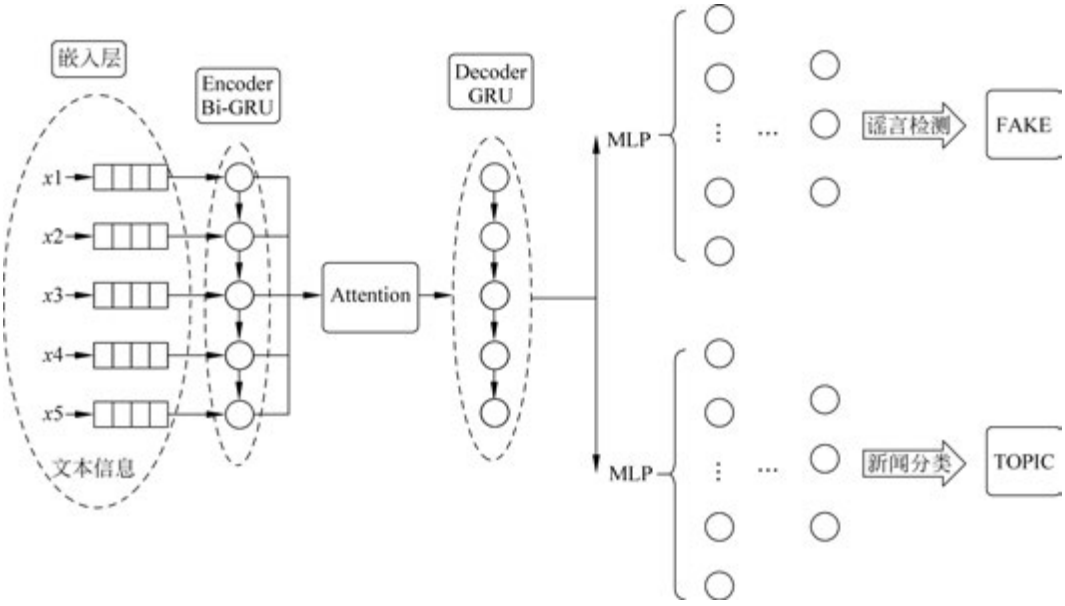

图 3-7　谣言检测的多任务模型

图 3-7 中，使用双向 RNN 结构处理输入的句子文本，前向和后向的基本单元均为 GRU，获得的输出序列进一步传递给 Attention 结构与后续的结构。具体可以使用 BahdanauAttention 等注意力机制，解码器采用单向 RNN 结构，结构的输出将分别输入两个无关联的 MLP 结构中，以实现下游两个分类任务的预测。以 GRU 为基本单元的 RNN 隐藏层的单元数为 1024。

GRU 能够对具有时间序列的数据进行特征学习，已经在众多自然语言处理任务中取得成功。通过不断地输入时间序列的数据，GRU 能够有选择地保留有用信息并将其作用于后续的计算中，从而获得各时间序列对象的高阶特征表示。

在深度学习中，注意力机制是通过对输入赋予不同的权重系数来实现的，权重系数越大则信息越值得被关注。注意力机制的计算过程可概括为

$$\text{Attention}(\text{Query},\text{Source})=\sum_{i=1}^{L}\text{Similarity}(\text{Query},\text{Key}_i)\times\text{Value}_i \tag{3-5}$$

其中，Source 由一系列的< Key，Value >数据对构成；L 为 Key 和 Value 的数量。当给定 Query 时，通过计算 Query 和每个 Key 的相似程度，可以获得每个 Key 对应 Value 的权

重系数,该权重系数则表示了 Value 的重要程度,最终的输出即为权重系数对 Value 的加权求和。

因此,注意力机制本质上是对 Value 值进行加权求和,而加权求和所用到的权重系数则由 Query 和 Key 进行相似性计算得到。在计算 Query 和 Key 的相似程度时,可以使用不同的计算方法。

3. 数据处理

由于 LIAR 数据集涵盖主题多达 140 个,并且一些主题对应的样本数目较少,因此在处理新闻的主题标签时,本节只取前 24 个出现次数最多的主题作为新闻的主题标签,其余的新闻则赋予 Others 标签。因此,对假新闻检测任务而言,是一个六分类问题;对新闻主题分类任务而言,是一个二十四分类问题。LIAR 数据集中涉及的主题包括 Economy、Health Care、Immigration、Crime 等。

主要的数据处理环节如下。

1) 文本表示

文本表示方法使用了 GloVe 技术。GloVe(Global Vectors for Word Representation)是一个基于全局词频统计(count-based & overall statistics)的词表征(word representation)工具,它可以把一个单词表达成一个由实数组成的向量,这些向量捕捉到了单词之间的一些语义特性,例如相似性(similarity)、类比性(analogy)等。

2) 文本清洗

针对文本内容进行必要的预处理,如去除停止词,替换数字、金钱面额、日期等对假新闻检测和新闻主题分类无关键意义的词;文本的最大长度为 30,文本长度超过 30 的截断后面的内容丢弃不用,文本长度不足 30 的用 0 填补。使用预训练的 GloVe 词向量初始化词嵌入矩阵,词向量的嵌入维度为 300,对于未登录词全零初始化。

3) 模型的训练

模型实现主要使用 TensorFlow 框架,代码使用 Python 编写。本节提出的多任务模型中的损失函数为两个任务的损失函数叠加。模型训练使用 Adam 算法,初始学习率设为 0.01,最大迭代次数设置为 10。

4. 主要代码

具体代码示例如下。

```
#谣言检测的 MLP 结构
class MLP_RUMOR(tf.keras.Model):
    def __init__(self):
        super(MLP_RUMOR, self).__init__()
        self.dense1 = tf.keras.layers.Dense(units = 256, activation = 'relu')
        self.dense2 = tf.keras.layers.Dense(units = 64, activation = 'relu')
        self.dense3 = tf.keras.layers.Dense(units = 16, activation = 'relu')
        self.dense4 = tf.keras.layers.Dense(units = 6, activation = tf.keras.activations.
softmax)
```

```
        self.bn1 = tf.keras.layers.BatchNormalization()

    def call(self, x, training):
        x = self.bn1(x, training=training)
        x = self.dense1(x)
        x = self.dense2(x)
        x = self.dense3(x)
        x = self.dense4(x)
        return x
#新闻主题发现的 MLP 结构
class MLP_NEWS(tf.keras.Model):
    def __init__(self):
        super(MLP_NEWS, self).__init__()
        self.dense1 = tf.keras.layers.Dense(units=512, activation='relu')
        self.dense2 = tf.keras.layers.Dense(units=128, activation='relu')
        self.dense3 = tf.keras.layers.Dense(units=64, activation='relu')
        self.dense4 = tf.keras.layers.Dense(units=25, activation=tf.keras.
activations.softmax)
        self.bn1 = tf.keras.layers.BatchNormalization()

    def call(self, x, training):
        x = self.bn1(x, training=training)
        x = self.dense1(x)
        x = self.dense2(x)
        x = self.dense3(x)
        x = self.dense4(x)
        return x

#训练函数
def train_step(inp, targ_news, targ_rumor, enc_hidden):
    with tf.GradientTape() as tape:
        # encoder
        enc_output, enc_hidden = encoder(inp, enc_hidden)
        dec_hidden = enc_hidden

        #decoder
        for i in range(32):
            dec_hidden, _ = decoder(dec_hidden, enc_output)

        # mlp
        predictions_news = mlp_news(dec_hidden, training=True)
        predictions_rumor = mlp_rumor(dec_hidden, training=True)

        # 损失函数定义为两个任务损失的平均
        batch_loss = 1/2 * (tf.reduce_mean(loss(targ_news, predictions_news)) +
tf.reduce_mean(loss(targ_rumor, predictions_rumor)))

    # acc
```

```
        batch_news_acc = acc_func(predictions_news, targ_news)
        batch_rumor_acc = acc_func(predictions_rumor, targ_rumor)

    #需要优化的参数包括编码器和解码器以及两个任务的 MLP 参数,通过 tf.GradientTape 梯度优
    #化器来求解
    variables = encoder.trainable_variables + decoder.trainable_variables \+
                mlp_news.trainable_variables + mlp_rumor.trainable_variables
    gradients = tape.gradient(batch_loss, variables)
    optimizer.apply_gradients(zip(gradients, variables))
    return batch_loss, batch_news_acc, batch_rumor_acc
```

5. 结果检验

为了验证多任务学习的有效性,需要把多任务学习模型与单任务谣言检测模型进行对比,考虑到简单的单任务模型存在过拟合风险,因此,在单任务模型上增加了使用dropout 方法的模型用来对比多任务学习模型的有效性。

从表 3-6 可以发现,多任务学习模型对分类结果的提升是显著的。其中,谣言检测任务的准确率从 19.3% 提升至 26.4%,最为明显。经过分析,我们认为因为单任务的谣言检测任务过于困难,模型很难学习到有用的特征,而构建多任务学习模型后,谣言检测模型因为和新闻分类模型共享底层结构,多任务的底层模型更好地学习到了有用的特征(两个任务共同作用,底层结构学习到了更加泛化的有用的特征),从而使得分类准确率得到提升。

表 3-6 模型在新闻数据集上的准确率

模　　型	谣 言 检 测	新 闻 分 类
单任务	19.3%	49.5%
多任务学习	26.4%	50.6%
dropout	20.5%	49.0%

新闻分类任务的准确率从 49.5%提升至 50.6%。相比谣言检测任务,新闻分类任务虽也有提升,但提升幅度较小。此外可以看出,单任务的新闻分类方法的准确率比单任务的谣言检测方法的准确率高很多,可以认为新闻分类任务更能学习到好的特征。因此,相比于谣言检测任务对新闻分类任务的作用,后者对前者的帮助要更显著一些,这也体现了多任务学习的主要特征。

实验中发现单任务的新闻分类任务在训练集上的准确率很高,而在测试集上的准确率要低很多,因此使用 dropout 对单任务模型进行正则化,并作为对比实验。结果表明,dropout 层对模型没有提升效果或效果很不明显。

William Yang Wang 在该数据集上的测试[2],得到的性能报告如表 3-7 所示。其中,使用 CNN 文本分类方法,并且要求提供新闻的额外属性,如主题词、作者等。而本节的多任务学习只使用文本信息作为输入,就能获得与利用多个属性的模型相似的效果。

表 3-7 虚假新闻检测

模 型	测试集的性能	模 型	测试集的性能
SVM	25.5%	混合 CNN：文本＋所在州	25.6%
Bi-LSTM	23.3%	混合 CNN：文本＋上下文	24.3%
混合 CNN：文本＋主题词	23.5%	混合 CNN：文本＋历史信用	24.1%
混合 CNN：文本＋作者	24.8%	混合 CNN：文本＋所有	27.4%

3.3.5 有待人工智能解决的问题

虚假新闻检测是一个实际的网络空间安全问题，对于人工智能技术也提出了很大的挑战。从当前研究及今后进一步发展来看，其挑战性主要体现在以下几方面。

1. 多模态信息的综合利用

新闻信息中除了文本信息外，还存在图片、视频等其他模态信息，同时，虚假新闻在社交网络中的传播方面也会体现出与正常新闻不同的行为特征，因此，社交网络中蕴含的多种特征也为多模态信息提供了有益补充。

2. 在机器学习中理解和运用虚假新闻的产生意图

虚假新闻的产生显然存在有别于正常新闻的原因，这些原因来自个体层面和社会层面。个体无法准确地区分真假新闻，达克效应、确认偏差、规范影响理论等社会理论决定个体分享与他们认知一致的信息的个性需求。社会层面的回声室效应、社会趋同性(homophily)、算法个性化推荐容易导致个体不愿意过多产生冲突观点。

3. 面向细分虚假信息的机器学习模型

虚假信息实际上可以细分为很多类型，例如伪造内容、误导性内容、冒名顶替内容、恶意内容、恶作剧、讽刺等，而这些类型有一些相同特征，但也体现出一定差异。如何让智能技术对虚假新闻内容做更进一步的判断，对于虚假新闻的引导和界定是有益的。但目前限制于数据样本和语义理解技术，还无法构建这些类型的机器学习模型。

参考文献

[1] Kar D, Panigrahi S, Sundararajan S. SQLiGoT: detecting SQL injection attacks using graph of token and SVM[J]. Computers & Security, 2016, 60(7): 206-225.

[2] Wang W Y. Liar, Liar Pants on Fire: a new benchmark dataset for fake news detection[C]. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, 2017: 422-426.