西北工業大學
NORTHWESTERN POLYTECHNICAL UNIVERSITY

第十一章　智能软件测试技术

授课教师：　郑炜

第十一章 智能软件测试技术

11.1 基于机器学习的软件缺陷报告分析

11.1.1 基于机器学习的软件缺陷报告分析的目的和意义

11.1.2 安全缺陷检测数据集的构造

11.1.3 基于深度学习的安全缺陷报告检测

11.2 基于软件度量的软件缺陷预测方法

11.2.1 基于软件度量的软件缺陷预测的目的和意义

11.2.2 软件缺陷预测模型的构建

11.2.3 软件缺陷预测模型性能评估指标

11.2.4 软件缺陷预测常用评测数据集

11.3 基于搜索的软件测试方法

11.3.1 智能搜索算法

11.3.2 搜索技术在软件测试中的应用

11.1.1 基于机器学习的软件缺陷报告分析的目的和意义

软件缺陷报告是软件设计、开发、维护过程中发现的软件问题的记录，是测试人员和开发人员之间重要的沟通工具。

常见的软件缺陷报告管理过程如图11-1所示，该过程包括重复检测、设置严重性和优先级、分配合适的开发人员、软件缺陷定位、软件缺陷修复，以及回归验证这几个关键步骤。

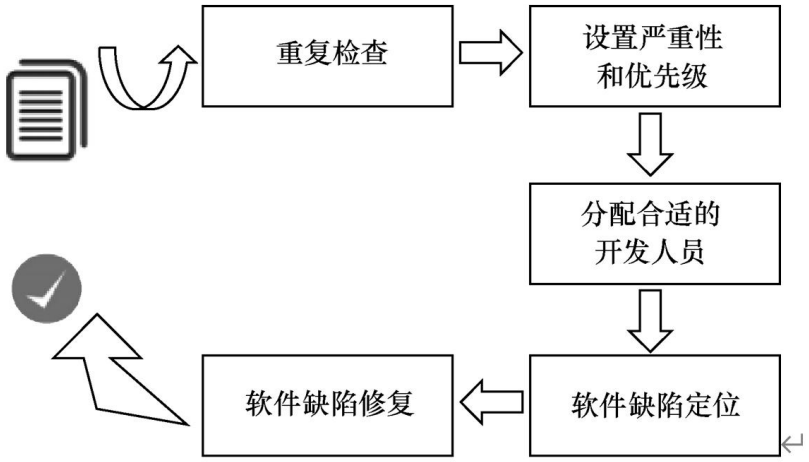

图11-1　软件缺陷报告管理过程

11.1.1 基于机器学习的软件缺陷报告分析的目的和意义

为了提高软件缺陷报告管理效率、降低软件缺陷管理成本，需要进行自动化软件缺陷报告分析。

1．自动化软件缺陷报告分析涉及的主要方面

针对软件缺陷报告的一系列分析研究，主要涉及以下5个方面。

(1) 重复软件缺陷报告检测

(2) 高影响力软件缺陷报告识别

(3) 软件缺陷报告优先级和严重性预测

(4) 软件缺陷/故障定位

(5) 自动程序修复

2．安全缺陷报告分析的背景

互联网的快速发展导致各种网络安全事件频繁发生。软件因为自身存在缺陷而遭受外界利用攻击导致隐私泄露、非法提权、勒索等严重后果，从而给个人、企业、社会和国家造成严重威胁。

研究表明，互联网面临的安全攻击（如拒绝服务攻击、缓冲区溢出、SQL 注入、病毒、数据篡改等）几乎都是利用软件中存在的安全漏洞实施的。

11.1.1 基于机器学习的软件缺陷报告分析的目的和意义

图11-2所示的是从2010年到2019年CVE网站每年新增的漏洞数量。从2010年开始每年至少新增4000条记录，特别地，从2017年开始数量出现猛烈增长，近三年（2017年—2019年）产生数据总量为43444条记录，占近十年（2010年—2019年）数据总量的52%。

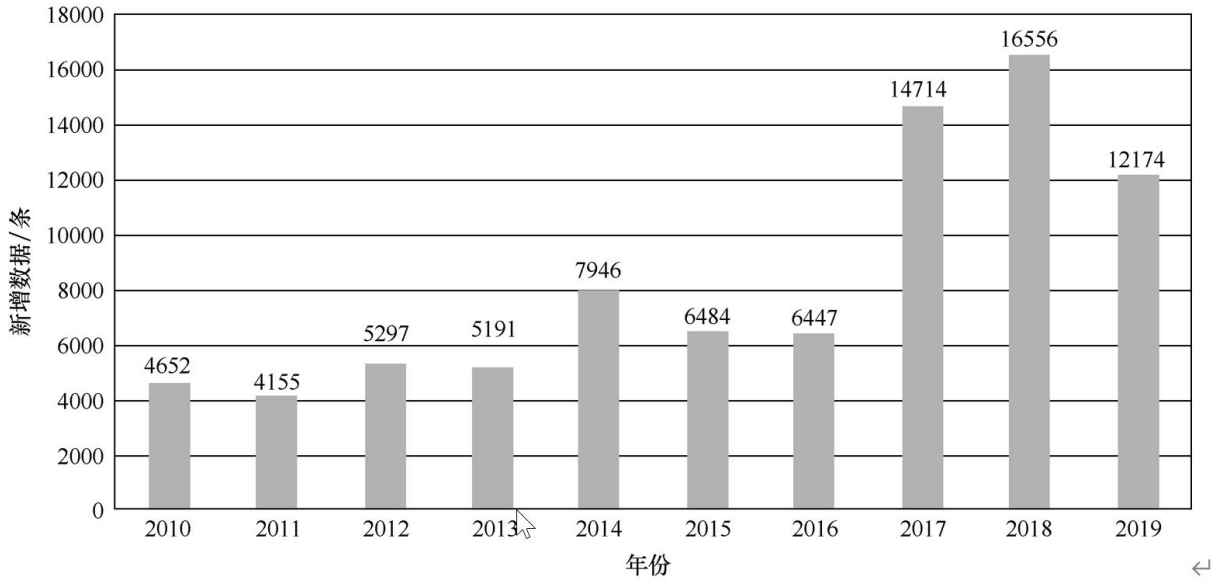

图11-2　2010年至2019年CVE数据增长趋势

11.1.2 安全缺陷检测数据集的构造

数据是机器学习应用的基础，基于机器学习的软件缺陷报告分析，首先需要有大规模带标记的软件缺陷报告数据的支持。

用于安全缺陷检测的数据集构造主要以开源的软件缺陷报告数据仓库（如JIRA、Bugzilla、LaunchPad）和开源漏洞数据库（如CVE、NVD、SARD）为数据源。

1．安全漏洞数据库CVE

对于开源项目，CVE详细信息页面的“参考”部分列出了相关源软件缺陷报告的链接，这些链接连接了CVE条目和源软件缺陷报告，因此，为安全缺陷检测数据集的构造提供了极大的便利。

2．人工样本标记

当前，在软件工程领域，许多样本数据的标记依然在很大程度上依赖领域专家知识，主要靠人工标记的方式来完成，面向安全缺陷报告分析的样本标记亦是如此。

安全缺陷报告检测自动化过程中，通常将安全缺陷报告检测形式化为一个二分类问题。即，与安全相关的软件缺陷报告（Security Bug Report，SBR），标记为“1”；与安全不相关的缺陷报告（Non-Security Bug Report，NSBR），标记为“0”。

11.1.2 安全缺陷检测数据集的构造

通常采用“卡片分类法”来进行样本标记，并通过Fleiss Kappa系数衡量不同成员标记结果之间的差异性。

卡片分类法：

组织至少两名成员对样本数据进行独立标记；然后，根据各成员标记结果判断最终样本标签，如果不同成员的标记结果一致，则该标签为该样本的最终标签；如果各成员标记结果不同，则成员进行沟通讨论，直到达成一致标记结果为止。

Fleiss Kappa系数是检验实验标注结果数据一致性的一个重要指标。Fleiss Kappa对于评定条件属性与条件属性之间的相关程度有很大帮助。

11.1.2 安全缺陷检测数据集的构造

设N为被评定对象的总数，n为评定对象的总数，k为评定的等级数，nij为第j个评定对象对第i个被评对象划分的等级数，则系数的计算公式为：

$$\overline{P} = \frac{1}{N}\sum_{i=1}^{N} P_i$$

$$\overline{P}_e = \sum_{j=1}^{k} P_j^2$$

公式（11.1）

其中

$$P_i = \frac{(\sum_{j=1}^{N} n_{ij}^2) - N}{N \times (N-1)}, P_j = \frac{\sum_{j=1}^{k} n_{ij}}{N \times n}$$

公式（11.2）

3．自动化样本标记预处理

面向软件缺陷报告分析的自动化样本标记方法的相关研究目前尚处于初期探索阶段。具体的自动化样本标记预处理过程主要涉及数据析取、数据清洗及数据融合3个主要阶段。

（1）数据析取

首先从软件缺陷跟踪系统（如JIRA、Bugzilla、LaunchPad）获取软件项目的缺陷报告数据极其详细信息，一般而言，一个软件缺陷报告主要包括：ID、标题（Title）、提交人员（Reporter）、软件缺陷修复人员（Assignee）、状态（Status）、优先级（Priority）严重程度（Severity/Importance）以及描述信息（Description）等字段。

11.1.2 安全缺陷检测数据集的构造

表11-1所示的是OpenStack项目中一个软件缺陷报告的具体示例。

表 11-1　来自 OpenStack 项目中的一个软件缺陷报告示例

字段	内容
ID	1713671
Title	Google Chat Screen Share Bug
Reporter	izbyshev
Assignee	Unassigned
Status	UNCONFIRMED
Priority	P2
Severity	S3

11.1.2 安全缺陷检测数据集的构造

字段	内容
Description	**User Agent**: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.212 Safari/537.36 **Steps to reproduce**: Load Google Chat and share screen (3 monitor setup - "Monitor 1") Move Firefox browser window to that screen, and open Google Slides presentation in the Firefox browser while screen sharing it Hit the black "Present" button and the browser goes to fullscreen mode Escape full screen mode by hitting "ESC" on the keyboard **Actual results**: After escaping or backing out of the fullscreen browser window, participants on the Google Chat call see the shared screen as only a portion of the actual shared visual area, and it appears as if other open Firefox tabs are behind or interfering with the shared screen contents. With further testing with a colleague of mine, it seems like fullscreen mode on Google Slides in the browser triggers the issue while screen sharing with Firefox as the active browser on the shared screen. **Expected results**: What should have happened: The shared Google Slides presentation should exit fullscreen mode and the shared window session should be refreshed properly to preview conflicting tabs/applications/windows.

11.1.2 安全缺陷检测数据集的构造

自动化样本标记过程一般需要借助一部分已经标记的样本信息。对于安全缺陷而言，CVE是当前国际权威的漏洞数据库，提交到 CVE 的软件缺陷可以认为都是跟安全相关的缺陷，因此，用户可以借助CVE进行一部分的安全缺陷样本标记。

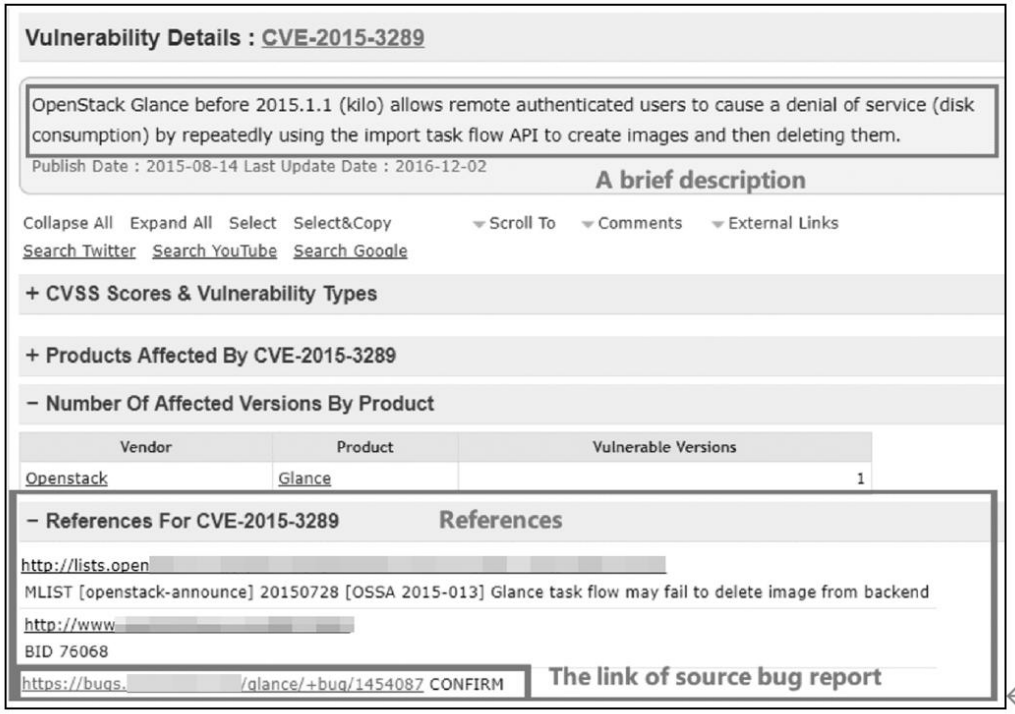

图11-3 CVE数据示例

（2）数据清洗

软件缺陷报告分析往往以描述信息作为关键分析内容，因此描述信息量过少，甚至为空的软件缺陷报告往往不适合进行自动化软件缺陷报告分析。数据清洗旨在对数据进行结构化和归一化等操作，过滤掉存在关键信息缺失等现象的无效数据。

（3）数据融合

自动关联多源数据之间的关系，例如，图11-3所示的是CVE数据与原始软件缺陷报告建立管理关系，从而获取到该软件缺陷报告的标记类型为安全缺陷报告。数据融合中获取的已标记样本数据将为后续自动样本标记模型的设计和构建提供样本数据基础。

4．面向安全缺陷报告检测的自动化样本标记方法

与人工标记过程类似，自动化样本标记中，依然采用SBR表示安全相关缺陷报告，标记为“1”；用NSBR表示与安全不相关的缺陷报告，标记为“0”。

（1）OpenStack简介

OpenStack是一个由美国国家航空航天局（National Aeronautics and Space Administration，NASA）和Rackspace公司合作发起的开源云管理平台，该平台包含多个用于构建云平台的组件（如Nova、Neutron、Swift、Cinder 等）。

（2）基于迭代式投票方法的自动化样本标记

图11-4所示的为迭代式自动化样本标记方法示意图，其具体过程分为两个阶段：初始数据准备和迭代式投票分类。

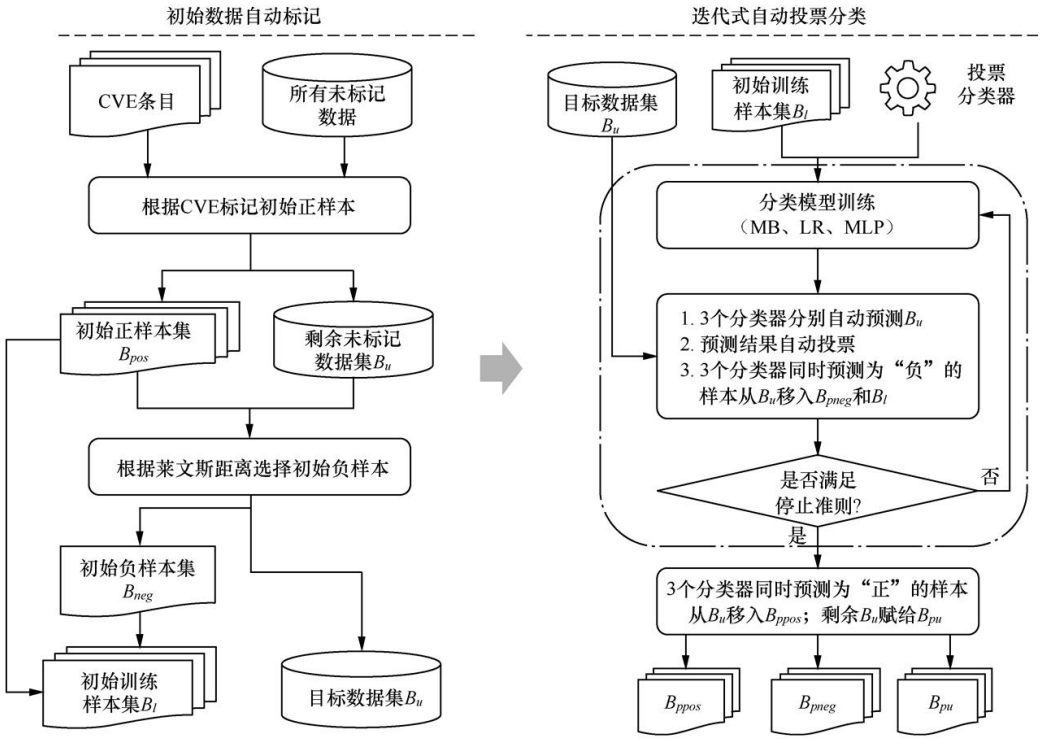

图11-4　迭代投票机制的自动化样本标记方法

11.1.2 安全缺陷检测数据集的构造

基于迭代式投票方法的自动化样本标记算法的具体步骤描述如下。

步骤1：初始训练样本标记。该阶段包括两个输入和两个输出。

输入：CVE数据和未标记样本Ball；

输出：已标记初始训练样本集Bl和剩余未标记样本集Bu。

步骤2：迭代投票式自动分类。基于软件缺陷报告中安全漏洞相关软件缺陷报告分布稀疏特征，利用3个不同文本分类器，设计了一种迭代式自动投票分类方法，其包括3个输入和3个输出。

输入：已标记训练样本集Bl、未标记目标数据集Bu、3个分类器。

输出：预测为正的样本集合Bppos、预测为负的样本集合Bpneg、不确定样本集合Bpu

11.1.2 安全缺陷检测数据集的构造

最终输出结果如表11-3所示。预测结果为负样本Bppos、Bpneg和Bpu所占比例分别为0.15%、99.27%和0.58%。

表 11-3　目标数据集迭代投票分类结果统计

数据集	B_{ppos}	比例（%）	B_{pneg}	比例（%）	B_{pu}	比例（%）
OpenStack	129	0.15	88146	99.27	515	0.58

基于深度学习的SBR预测方法框架如图11-5所示，其包含如下4个阶段。

阶段1：执行BR数据集的收集与划分。

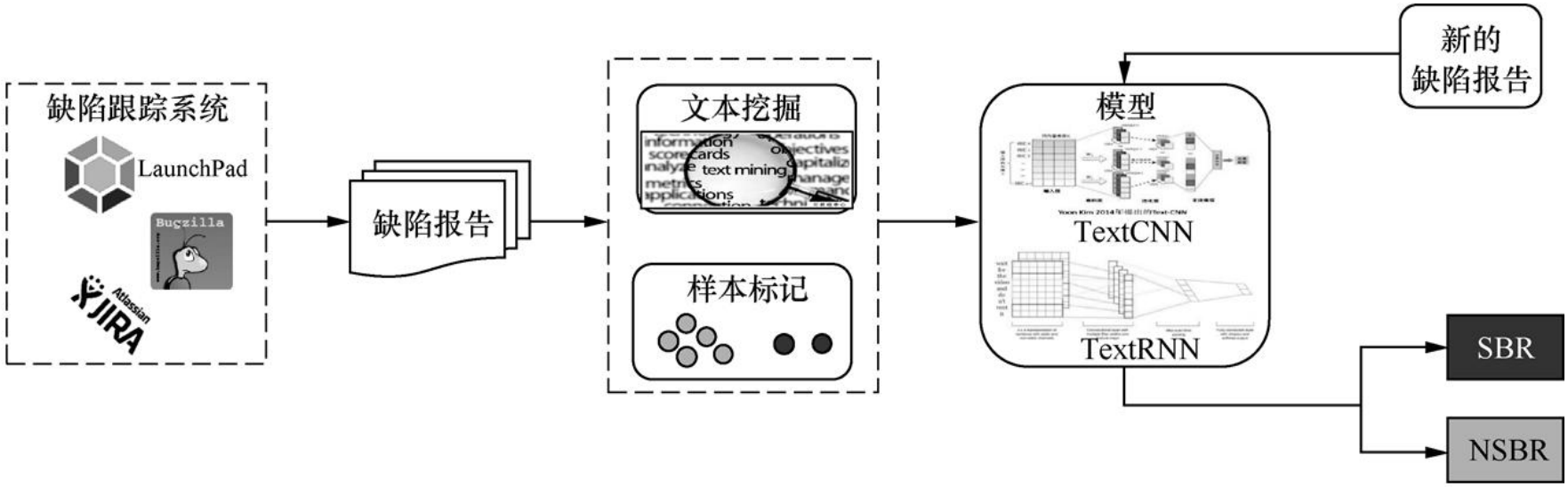

图11-5　基于深度学习的安全缺陷报告预测方法框架

阶段2：执行文本预处理。该阶段将文本数据预处理成数字序列，深度学习模型可以方便地根据序列获得单词的表示（词向量）。

阶段 3：进行深度学习模型的构建。我们一共选择了两种适合文本挖掘的深度学习模型，分别是TextCNN和TextRNN。这两个模型分别衍生自经典的CNN模型和RNN模型，选用这两个具有代表性的模型可以确保我们实证研究的结论更有代表性。

阶段4：目标数据预测与输出。从未标记的软件缺陷报告中预测中SBR，并对结果进行输出。

1．文本预处理

文本预处理时依次执行移除停顿词、句子填充、建立词汇—索引映射表和建立词嵌入矩阵。

（1）移除停顿词

（2）句子填充

（3）建立词汇—索引映射表

（4）建立词嵌入矩阵

2．深度学习模型构建

关于深度学习模型构建，下面主要考虑了两种深度学习模型TextCNN和TextRNN，并采用注意力机制对模型进行优化。

（1）TextCNN模型

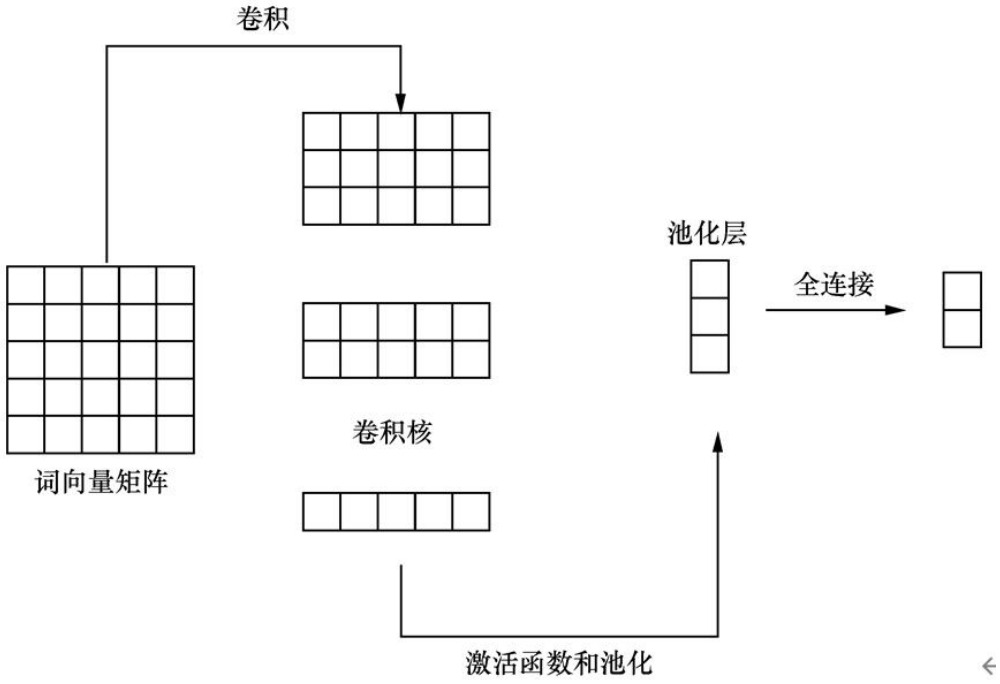

图11-6　TextCNN网络结构

(2) TextRNN模型

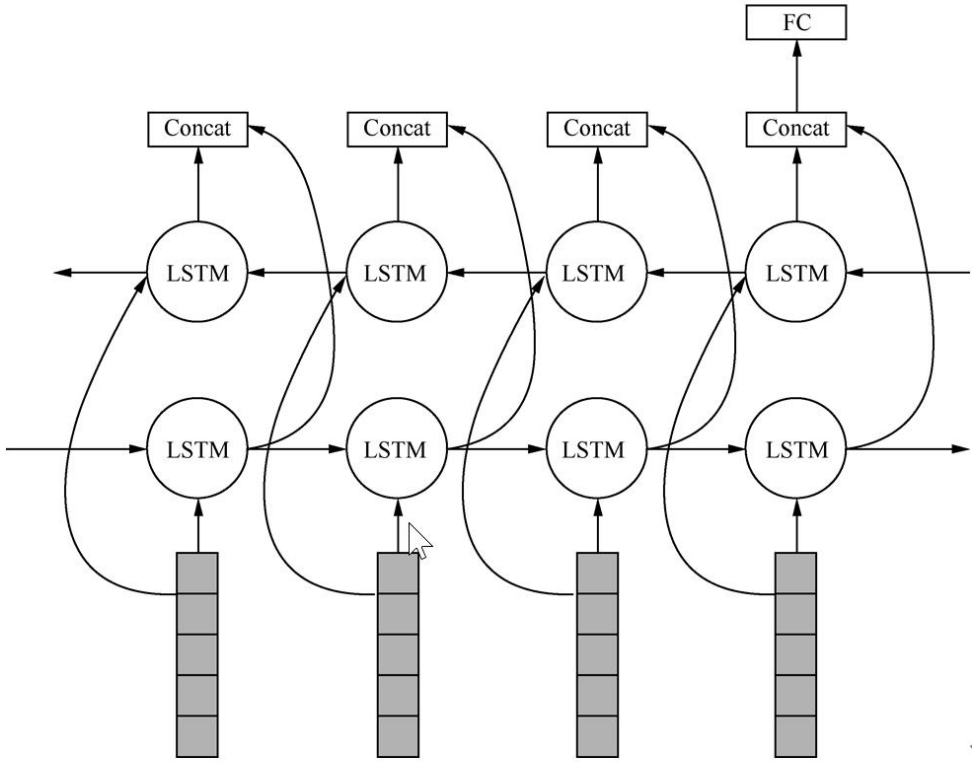

图11-7 TextRNN的结构图

（3）注意力机制

进一步借助注意力机制（Attention Mechanism）对TextRNN模型进行优化。SBR的识别主要依赖于BR中的描述信息（Description），采用自然语言描述形式，其安全相关文本特征较为稀疏，给SBR预测准确性带来一定困难。

注意力机制源于对人类视觉的研究，使训练重点集中在输入数据的相关部分，而忽略无关部分。

(4) 深度学习模型有效性

为了验证深度学习模型对于安全缺陷报告检测的有效性，采用公开数据集Ambari、Camel、Derby、Wicket和OpenStack对其进行验证，通过与传统分类算法朴素贝叶斯（Naive Bayes，NB）、逻辑回归（Logistic Regression，LR）、K-近邻算法（K-Nearest Neighbor）进行对比。

表 11-4　传统分类算法和深度学习方法的最优 F1-score 对比

数据集名称	传统分类算法		深度学习方法		F1-score 提升
	算法名称	F1-score	算法名称	F1-score	
Ambari	NB	0.125	Attention+TextRNN	0.615	0.490
Camel	LR	0.108	TextCNN	0.643	0.535
Derby	NB	0.360	Attention+TextRNN	0.464	0.104
Wicket	KNN	0.048	TextCNN 或 Attention+TextRNN	0.000	−0.048
OpenStack	NB	0.201	Attention+TextRNN	0.410	0.209

11.1 基于机器学习的软件缺陷报告分析

11.2 基于软件度量的软件缺陷预测方法

11.2.1 基于软件度量的软件缺陷预测的目的和意义

11.2.2 软件缺陷预测模型的构建

11.2.1 基于软件度量的软件缺陷预测的目的和意义

软件缺陷预测属于当前软件工程数据挖掘领域中的一个重要研究问题。随着新的数据挖掘技术的不断涌现，以及研究人员对软件历史仓库挖掘的日益深入，软件缺陷预测研究在近些年来取得了大量的研究成果，其软件缺陷预测结果也逐渐成为判断一个系统是否可以交付使用的重要依据。

因此针对该问题的深入研究，对提高和保障软件产品的质量具有重要的研究意义。

11.2.1 基于软件度量的软件缺陷预测的目的和意义

软件缺陷预测可以将程序模块的缺陷倾向性、缺陷密度或缺陷数设置为预测目标。以预测模块的缺陷倾向性为例，其典型框架如图11-8所示。

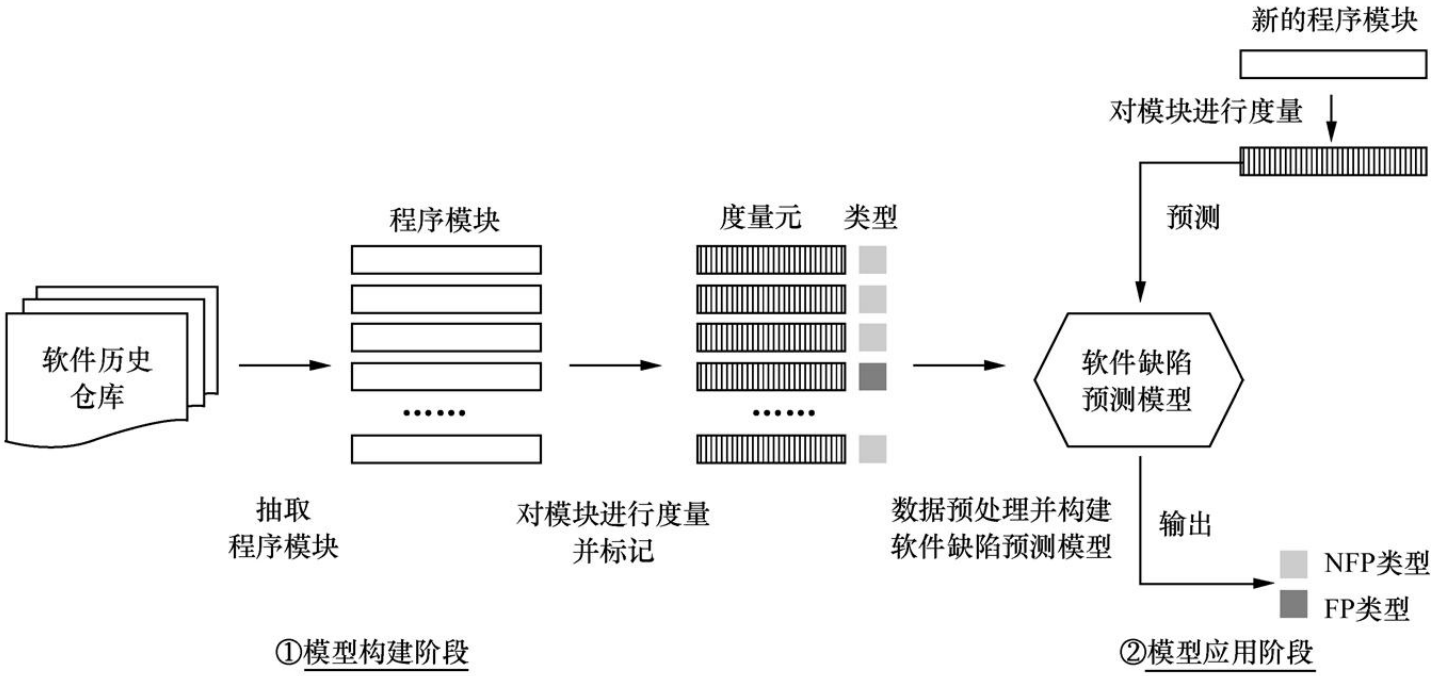

图11-8 以缺陷倾向性为预测目标的软件缺陷预测方法整体框架

11.2.1 基于软件度量的软件缺陷预测的目的和意义

该过程主要包括两个阶段：模型构建阶段和模型应用阶段。具体来说，模型构建阶段包括以下3个步骤。

① 挖掘软件历史仓库，从中抽取程序模块。

② 通过分析软件代码或开发过程设计出与软件缺陷存在相关性的度量元，借助这些度量元对程序模块进行软件度量并构建出软件缺陷预测数据集。

③ 对软件缺陷预测数据集进行必要的数据预处理（如噪声移除、特征子集选择、数据归一化等）后，借助特定的建模方法构建出软件缺陷预测模型。

11.2.1 基于软件度量的软件缺陷预测的目的和意义

1．对程序模块的度量

随着面向对象开发方法的普及，面向对象开发方法特有的封装、继承和多态等特性给传统的软件度量提出了挑战，于是研究人员提出了适用于面向对象程序的度量元，其中最为典型的是Chidamber和Kemerer提出的CK度量元。

表 11-5　CK 度量元

名称	描述
WMC	类的加权方法数
DIT	类在继承树中的深度
NOC	类在继承树中的孩子节点数
CBO	与该类存在耦合关系的其他类的数量
RFC	该类可以调用的外部方法数
LCOM	类内访问一个或多个属性的方法数

2．软件缺陷预测模型的构建方法

目前大部分研究工作都基于机器学习的方法来构建软件缺陷预测模型。其中常见的模型预测目标可以大致分为两类：一类是预测程序模块内含有的缺陷数或缺陷密度；另一类是预测程序模块的缺陷倾向性。

有研究人员设计出了BugCache方法，该方法主要考虑了如下的软件缺陷局部性原理：

① 修改模块局部性：即一个程序模块，若最近被修改过，则将来可能还会含有软件缺陷。

② 新增模块局部性：即一个程序模块，若是最近新增加的，则将来可能还会含有软件缺陷。

③ 时间局部性：即一个程序模块，若最近检测出一个软件缺陷，则将来可能也会含有其他软件缺陷。

④ 空间局部性：即一个程序模块，若最近检测出一个软件缺陷，则将来它附近的程序模块可能也会含有软件缺陷。

3．软件缺陷预测数据集的质量问题

（1）数据集内的噪声问题

（2）数据集内的维数灾难问题

（3）数据集中的类不平衡问题

4．跨项目软件缺陷预测方法

基于已有跨项目软件缺陷预测的研究成果分析，可以简单总结出如下3个场景。

① 仅存在与目标项目T相关的候选源项目的集合S={S_1,S_2,…,S_n}。其中T对应的数据集为t，候选源项目S_i对应的数据集为s_i。

② 不仅存在与目标项目T相关的候选源项目的集合S，而且在T中存在少量已标记的目标项目模块，并将这些已标记的模块记为tl。

③ 难以找到与目标项目T相关的候选源项目，同时目标项目中也不存在已标记的模块。

11.2.2 软件缺陷预测模型的构建

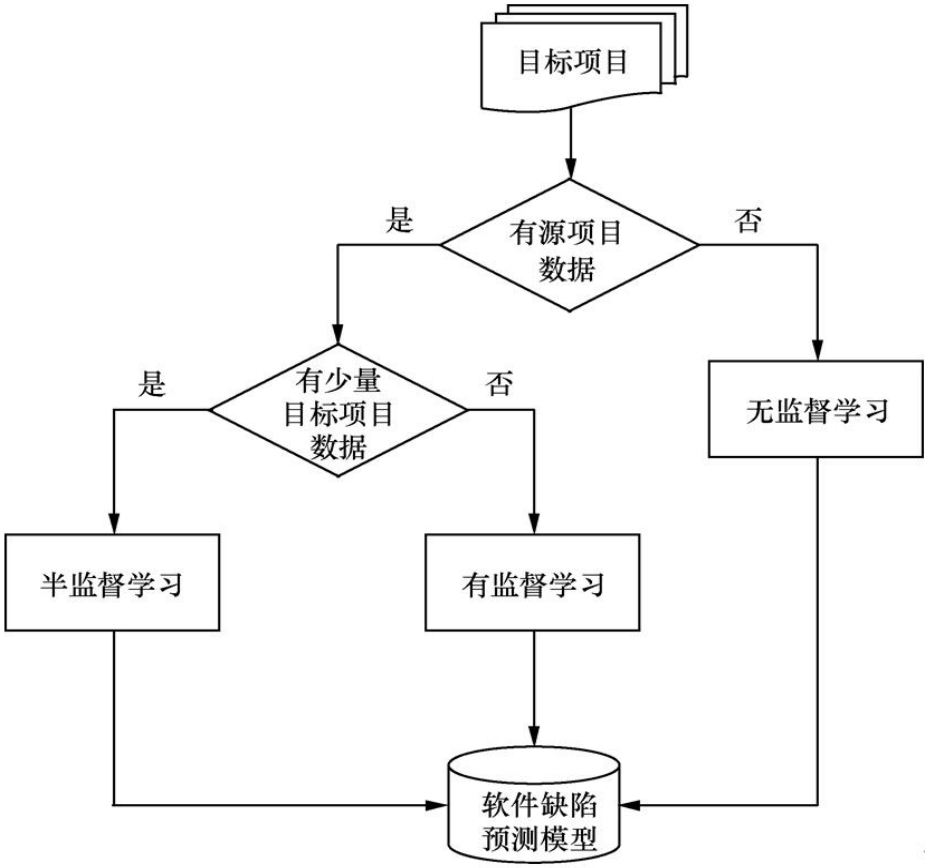

图11-10 跨项目软件缺陷预测的研究框架

11.2.3 软件缺陷预测模型性能评估指标

表 11-6　混淆矩阵

真实类型	预测类型	
	有缺陷模块	无缺陷模块
有缺陷模块	TP	FN
无缺陷模块	FP	TN

11.2.3 软件缺陷预测模型性能评估指标

基于上述混淆矩阵，我们将已有研究工作中常用的模型性能评测指标总结如下。

（1）准确率（Accuracy）

（2）查准率（Precision）

（3）查全率（Recall）

（4）虚警概率

（5）F-measure

（6）G-measure

（7）Balance

（8）G-mean

（9）MCC

11.2.4 软件缺陷预测常用评测数据集

- NASA数据集
- PROMISE数据集
- AEEEM数据集
- Relink数据集

11.3 基于搜索的软件测试方法

11.3.1 智能搜索算法

11.3.2 搜索技术在软件测试中的应用

11.3.1 智能搜索算法

基于搜索的软件测试的基石是高性能的搜索算法，目前软件工程领域中用到的搜索方法分为3大类：

- 第1类是基于微积分的搜索方法
- 第2类是带有向导的随机搜索方法
- 第3类是枚举方法

11.3.1 智能搜索算法

主要介绍第2类带有向导的随机搜索方法：

1．遗传算法

遗传算法是一种模拟自然界生物进化过程的启发式搜索算法。它属于演化算法的一种，借鉴了进化生物学中的一些现象，这些现象通常包括遗传、突变、自然选择，以及杂交等。

遗传算法操作步骤中关键的几步操作为：从问题空间得到信息编码、染色体交叉，以及适应度函数的构造。

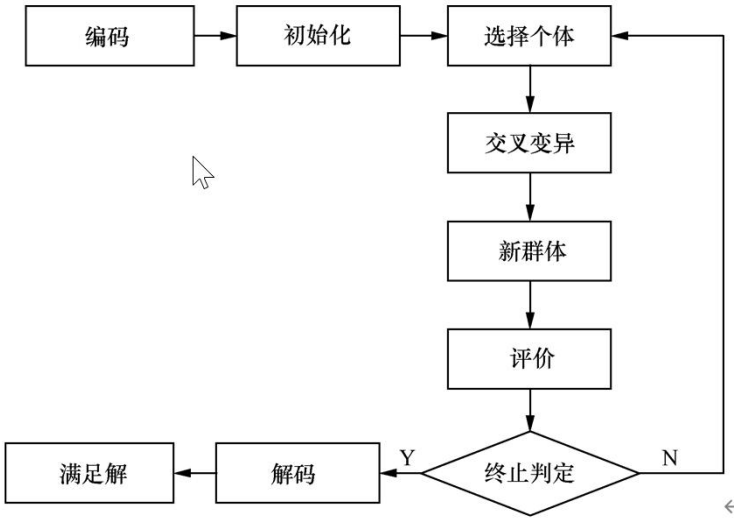

图11-13　遗传算法流程图

主要介绍第2类带有向导的随机搜索方法：

2．蚁群算法

蚁群算法的各个蚂蚁在没有事先被告诉食物在什么地方的前提下开始寻找食物，当一只蚂蚁找到食物以后，它会向环境释放一种挥发性分泌物信息素（Pheromone），该物质随着时间的推移会逐渐挥发消失。

信息素浓度的大小表征路径的远近，吸引其他的蚂蚁过来，这样越来越多的蚂蚁会找到食物。

主要介绍第2类带有向导的随机搜索方法：

2．蚁群算法

关于蚁群算法的几个比较重要的规则：

① 范围

② 环境

③ 觅食规则

④ 移动规则

⑤ 避障规则

⑥ 信息素规则

主要介绍第2类带有向导的随机搜索方法：

2．蚁群算法

蚁群算法有以下4个基本的特点：

① 蚁群算法是一种自组织的算法

② 蚁群算法是一种本质上并行的算法

③ 蚁群算法是一种正反馈的算法

④ 蚁群算法具有较强的鲁棒性

3．其他智能优化算法

(1) 人工鱼群算法

(2) 禁忌搜索算法

(3) 人工免疫算法

(4) 粒子群算法

(5) 爬山算法

(6) 模拟退火算法

1．基于搜索的测试用例自动生成

（1）功能测试用例生成方法

（2）结构测试用例生成方法

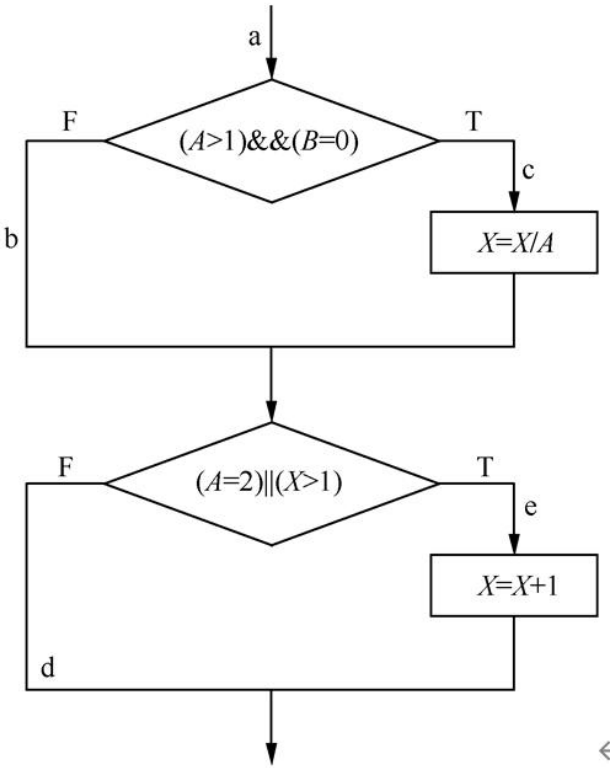

图11-16　程序段流程图

1．基于搜索的测试用例自动生成

（1）功能测试用例生成方法

功能测试也叫作黑盒测试，它不考虑程序内部的实现逻辑，以检验输入/输出信息是否符合规格说明书中有关功能需求的规定为目标。

（2）结构测试用例生成方法

结构测试也叫白盒测试，它主要检查程序的内部结构、逻辑、循环和路径。

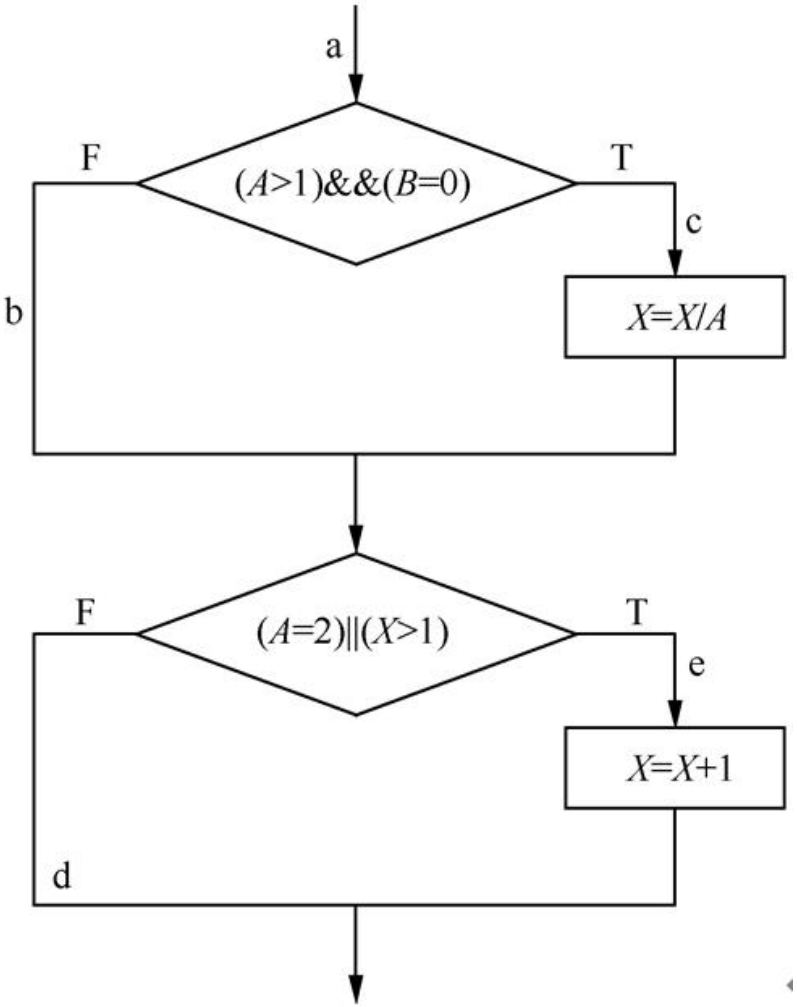

图11-16　程序段流程图

介绍逻辑覆盖和程序插装两种方法：

① 逻辑覆盖主要的测试用例设计方法有路径覆盖、语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、条件组合覆盖。

② 程序插装方法就是借助往被测程序中插入操作来实现测试目的的方法。

11.3.2 搜索技术在软件测试中的应用

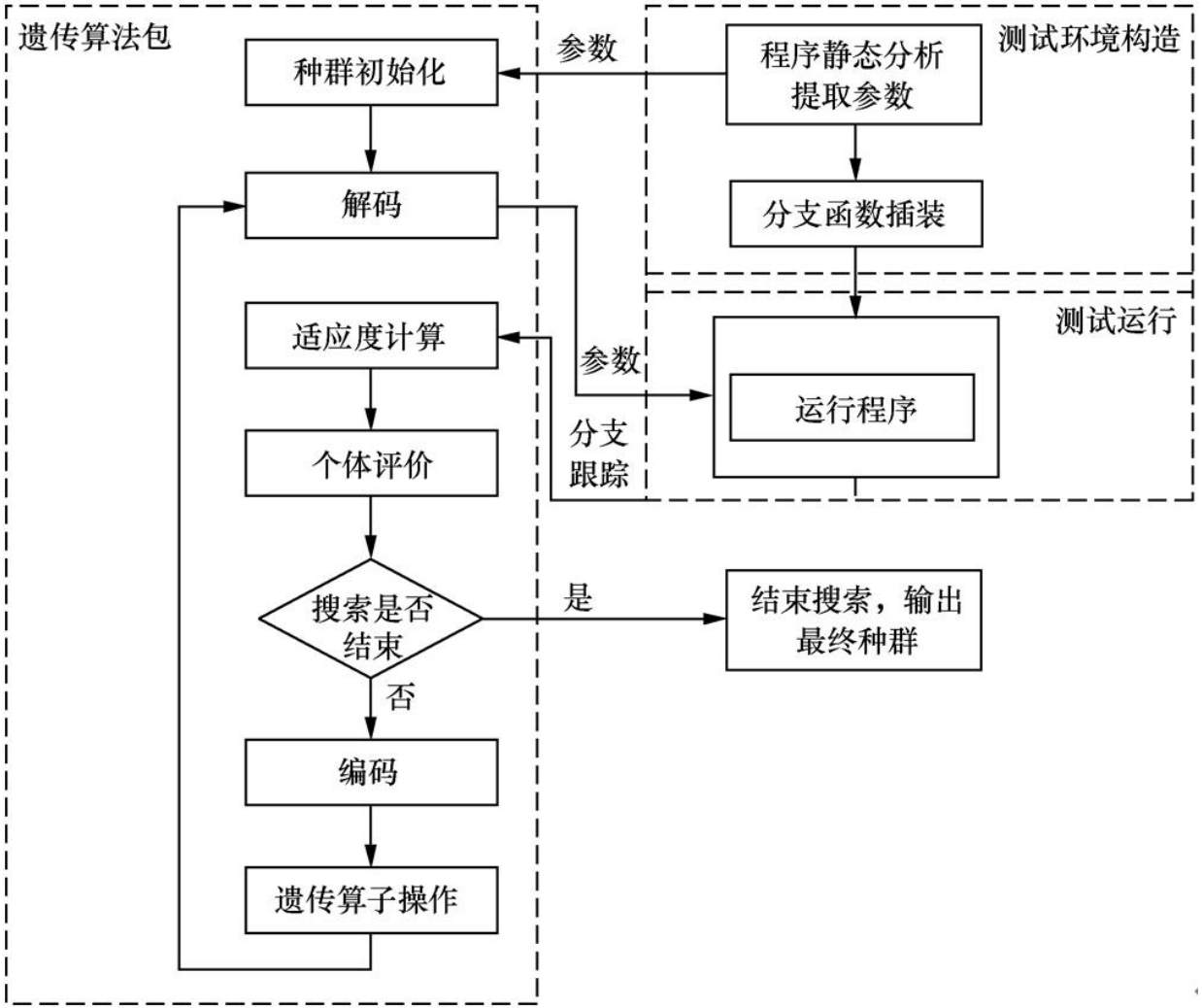

图11-17　测试用例生成系统模型图

2．基于搜索的测试用例优化

对于测试用例的优化主要是挑出覆盖率高和测试代价小的测试用例。

将讲解遗传算法在测试用例最小化的应用。

（1）测试用例最小化

（2）覆盖率数据库文件

（3）对测试用例建模

（4）遗传算法应用的具体实现过程

① 初始化编码：采用多参情况下的编码方案
② 计算个体适应度值
③ 选择：适应度越大表示这个个体越好
④ 交叉
⑤ 变异

3．基于搜索的变异测试技术

该方法选择了遗传算法，并结合面向对象程序的特征分别设定了染色体编码、适应值函数，以及变异、交叉和选择操作。

变异测试用例生成的本质就是求解杀死变异体的测试用例。这一求解过程要借助杀死变异体的 3个基本条件，现有的变异测试数据生成的方法是基于约束的生成方法和动态域削减方法。

本章小结

- 数据集的搜集过程、基于文本挖掘的软件缺陷报告预测模型的构建
- 程序模块度量方法、模型的构建方法和数据集质量问题
- 模型评测指标和经典的评测数据集
- 智能搜索算法，如遗传算法、爬山算法、模拟退火算法、粒子群优化算法