

# 第五讲 软件缺陷预测

---

李宁

2020.4

## ❑ 软件缺陷基础

- ✓ 软件缺陷概念
- ✓ 软件缺陷管理
- ✓ 软件缺陷分类体系
- ✓ 软件缺陷分析

## ❑ 软件可靠性

- ✓ 软件可靠性的定义
- ✓ 可靠性模型及其评价标准
- ✓ 软件可靠性测试

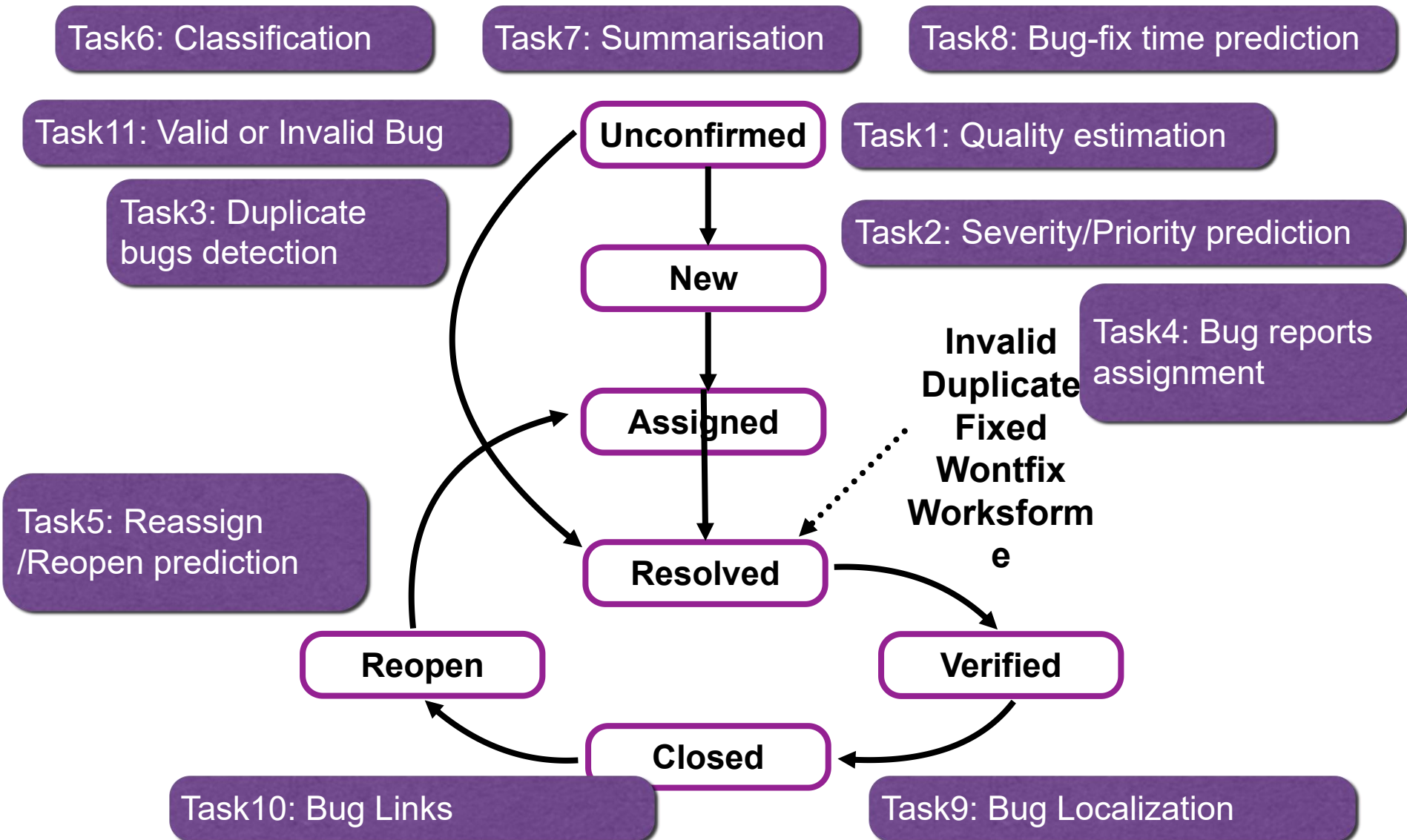
**1. 软件缺陷数据预处理**

**2. 软件缺陷分类**

**3. 软件缺陷预测**

**4. 软件缺陷智能分析**

# Lifecycle of a Bug Report (Xiaxin)



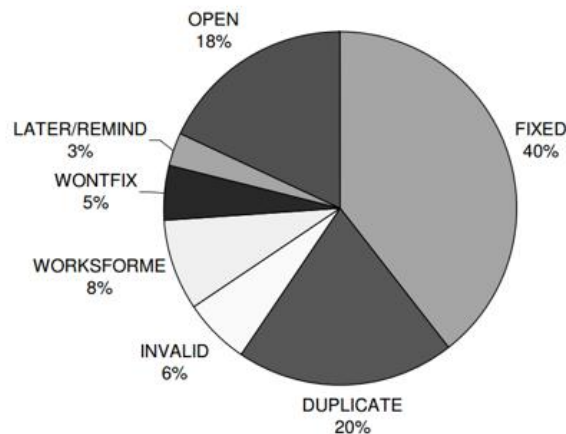
# 1 软件缺陷相关数据

## ■ 缺陷数据（缺陷报告）

- 重复数据（缺陷描述文本重复）
- 无效数据

## ■ 缺陷相关的软件项目数据

- 产品度量
- 项目度量
- 过程度量



(a) Types of bugs for Eclipse.

# 1 软件缺陷的去噪

## ■ 重复缺陷报告的去除

### ■ 重复缺陷报告的种类:

- ✓ 现象本身完全重复
- ✓ 现象不同，但却由同一原因导致  
(开发人员：重复，但测试人员：不重复)

解决方法：信息检索或者数据挖掘

# 1 软件缺陷的去噪

## ■ 其他无效缺陷报告的去除

### ■ 其他缺陷报告的主要种类:

- ✓ 无法重现
- ✓ 不是问题



判定的主观性较强，  
自动技术难以检测

### 相关的解决方案:

#### 1. 如何写一个好的缺陷报告的方法和工具?

- 缺陷报告的形式、内容等加以约束和检查实现  
(What makes a good bug report? TSE, 2010)

#### 2. 从缺陷报告中提取一些结构化的信息。

(Extracting structural information from bug reports, MSR2008)

# 1 软件缺陷的去噪

## ■ 去除重复缺陷报告的主要方法

- ✓ 文本相似度与图聚类
- ✓ 向量空间模型 (VSM)
- ✓ 自然语言处理 (NLP)

## ■ 去除重复缺陷报告所依据的主要数据

- ✓ 缺陷报告的标题文本
- ✓ 缺陷报告的正文文本
- ✓ 缺陷报告的执行信息



# 1 软件缺陷的去噪

## 1. 基于N-gram模型的重复缺陷报告检测技术

**N-gram:** 指文档中连续出现的N个项目构成的文本片段。  
此处的“项目”根据不同的应用可以是字母，单词或者词组。研究中，比较常见的是使用2-gram (Bigram) 和3-gram (Trigram) 作为特征。

例：某缺陷报告：Unable to print in PDF。

假设本例中的项目是单词，则：

**2-gram序列：**

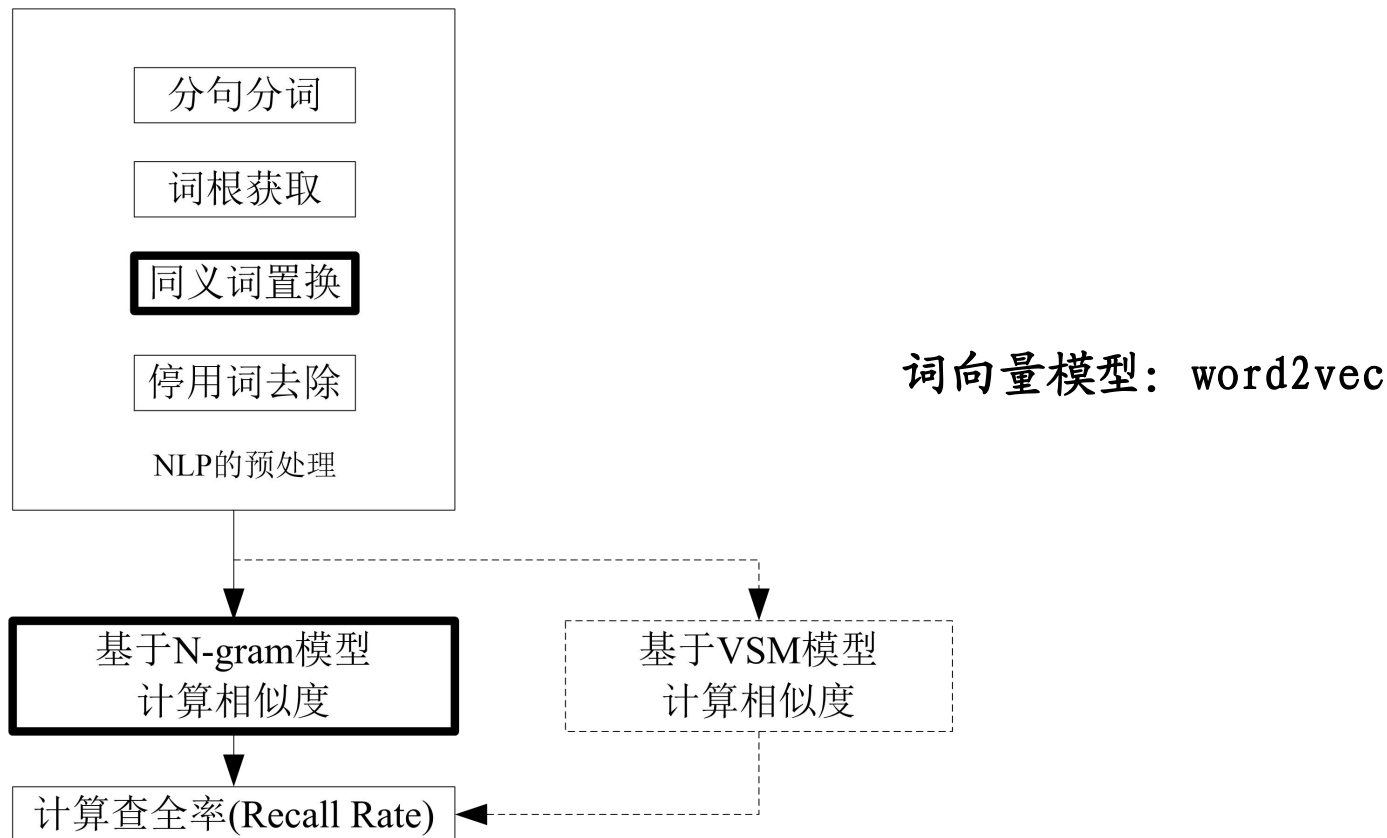
{Unable, Unable to, to print, print in, in PDF, PDF};

**3-gram序列：**

{Unable, Unable to, Unable to print, to print in, print in PDF, in PDF, PDF}。

# 1 软件缺陷的去噪

## ■ 基于N-gram模型的重复缺陷报告检测技术



<https://www.bilibili.com/video/BV1JJ411t7sU?from=search&seid=9198199485409910378>

# 1 软件缺陷的去噪

## ■ 基于N-gram模型的检测技术 - N-gram模型

设 $w_i$ 是文本中的任意一个项目，如果已知它在该文本中的前两个项目 $w_{i-2}w_{i-1}$ ，便可以用条件概率 $P(w_i|w_{i-2}w_{i-1})$ 预测 $w_i$ 出现的概率。

$$P(W) = P(w_1)P(w_2|w_1)P(w_3|w_1w_2)...P(w_n|w_1w_2...w_{n-1})$$

由于上式计算复杂，N-gram模型通常进行一种假设：第 $n$ 个项目的出现只与前面第 $n-1$ 个项目相关，而与其它任何项目都不相关。由此问题得到极大的简化，例如3-gram模型即为：

$$P(w_i|w_{i-2}w_{i-1}) \approx \text{count}(w_{i-2}w_{i-1}w_i) / \text{count}(w_{i-2}w_{i-1})$$

## ■ 基于N-gram模型的检测技术 - N-gram相似度

设N-gram(T)是字符串T中长度为N的子串的集合。两个字符串T1、T2的N-gram相似度Sim-NG(T1, T2)定义如下:

$$Sim - NG(T1, T2) = \frac{|N - gram(T1) \cap N - gram(T2)|}{|N - gram(T1) \cup N - gram(T2)|}$$

分母表示字符串T1、T2中所有的长度为N的子串的数目，  
分子表示字符串T1、T2中完全相同的长度为N的子串的数目。  
显然，Sim-NG(T1, T2)越大，字符串T1、T2越相似。

# 1 软件缺陷的去噪

## ■ 基于N-gram模型的检测技术 – 检测准确度度量

重复缺陷的**查全率**：在指定建议列表长度时，源缺陷报告 (Master Report) 的重复目标缺陷报告 (Target Report) 被正确检索到的比率。

$$RecallRate = \frac{N_{recalled}}{N_{total}}$$

Bug11
Bug12
Bug13
Bug14
Bug15

Master

Bug1
Bug2
Bug3
Bug4
Bug5
Bug6
Bug7
Bug8
Bug9
Bug10

Target

假定建议列表长度 (top list size) : 3

若仅Bug11, Bug12, Bug13, Bug14所列出的Top 3个Bug中有真正重复的缺陷 (Bug15的Top3中没有真正重的缺陷), 则:

查全率为:  $4/5=80\%$

查准率为:  $4/(5*3) = 27\%$

# 1 软件缺陷的去噪

## 2. 基于短语的N-gram法

卡耐基梅隆大学的Ko等人对软件缺陷报告的概述信息从语言学的角度进行了统计分析，认为**名词短语**、**介词短语**是最重要的。

尝试提出一种基于短语的N-gram法：

1. 抽取原始缺陷描述文本中的名词短语和介词短语。
2. 根据一定的规则重新组织上述短语，构成新的文本。
3. 在重组后的文本上利用N-gram法计算文本相似度。

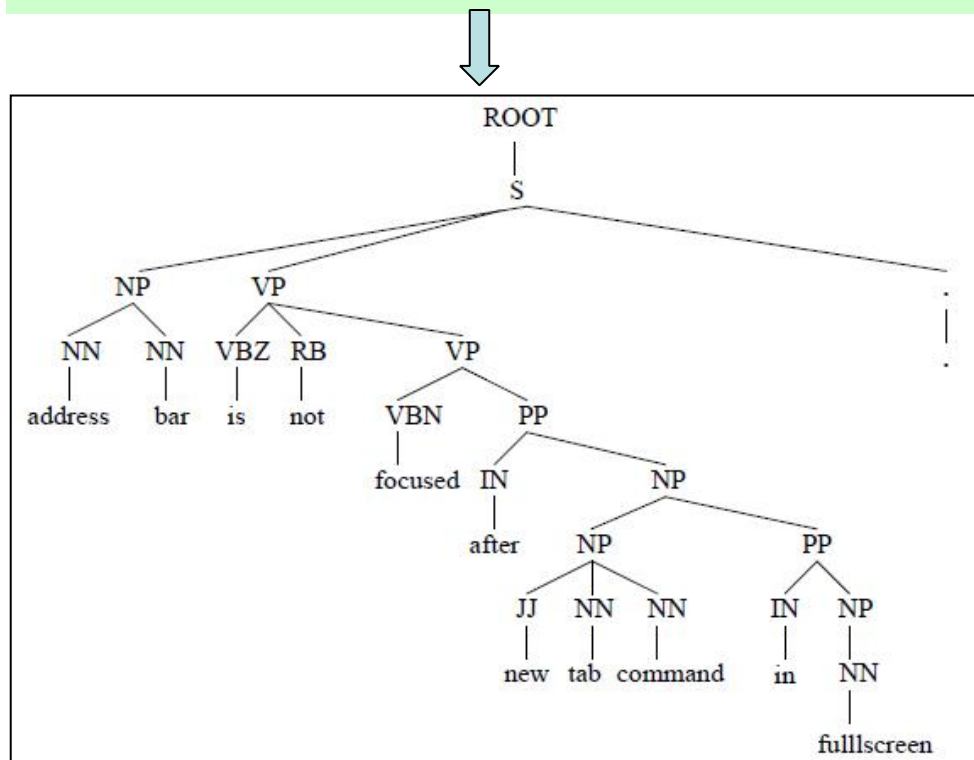
重组规则可以自己定义，例如：

1. NP中包含NP时，从长NP中删除短NP，短NP保持不变。
2. PP中包含PP时，从长PP中删除短PP，短PP保持不变。
3. PP中包含NP时，保留PP，删除NP。
4. NP中包含PP时，从NP中删除PP，PP保持不变。

# 1 软件缺陷的去噪

## ◆ 基于短语的N-gram法 - 语法解析

address bar is not focused after new tab command in fullscreen.



{NP}

- 1.address bar
- 2.new tab command in fullscreen
- 3.new tab command
- 4.fullscreen

{PP}

- 1.after new tab command in fullscreen
- 2.in fullscreen

重组规则

{NP}

- 1.address bar

{PP}

- 1.after new tab command
- 2.in fullscreen

address bar after new tab command in fullscreen

# 1 软件缺陷的去噪

## 3. VSM法

在文档向量空间模型中，每一个文本被看作是一个n维向量的形式：

$$d_j = \langle w_{1j}, w_{2j}, \dots, w_{tj} \rangle$$

- 1) 向量分量 $w_{ij}$  代表第 $i$ 个索引词在文档 $d_j$  中所具有的权重。
- 2)  $t$ 是系统中唯一索引词的个数。
- 3)  $w_{ij}$ 的取值范围是一个连续的实数区间 $[0, 1]$  。

索引词权值计算方案：tf-idf (inverse document frequency)

$$w_{ij} = tf_{ij} \times idf_i$$

$$tf_{ij} = \frac{freq_{ij}}{\max(tf_j)}$$

$$idf_i = \log\left(\frac{N}{n_i} + offset\right)$$

**tf:**表示局部权值，指第 $i$ 个索引词在第 $j$ 篇文档中的权重  
**idf:**表示全局权值，指第 $i$ 个索引词在整个文档集合中的权重

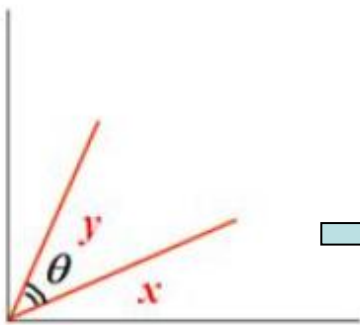
tf越大表明第 $i$ 个索引词在文档 $j$ 中越重要。

idf越小说明第 $i$ 个索引词越普遍，对文本区分的贡献度越小。



# 1 软件缺陷的去噪

## ■ VSM法 - 余弦定理计算两个向量的距离



在空间模型中，两条线的夹角越小，它们的余弦值就越大，而它们越相似(重叠或者平行)。

$$\cos(\theta) = \frac{T1 \cdot T2}{\|T1\| \|T2\|} = \frac{\sum_{i=1}^n W_{1i} W_{2i}}{\sqrt{\sum_{i=1}^n (W_{1i})^2} \times \sqrt{\sum_{i=1}^n (W_{2i})^2}}$$

例如:

X文档:  $\langle x_1, x_2, \dots, x_{64000} \rangle$

Y文档:  $\langle y_1, y_2, \dots, y_{64000} \rangle$

$$\cos \theta = \frac{x_1 y_1 + x_2 y_2 + \dots + x_{64000} y_{64000}}{\sqrt{x_1^2 + x_2^2 + \dots + x_{64000}^2} \cdot \sqrt{y_1^2 + y_2^2 + \dots + y_{64000}^2}}$$

# 1 软件缺陷的去噪

## ■ VSM法 – 余弦定理计算两个向量的距离

$$Sim - Cos(T1, T2) = cos(\theta) = \frac{T1 \cdot T2}{\|T1\| \|T2\|} = \frac{\sum_{i=1}^n W_{1i} W_{2i}}{\sqrt{\sum_{i=1}^n (W_{1i})^2 \times \sum_{i=1}^n (W_{2i})^2}}$$

### □ 例题：计算以下两个BugReport的相似度

Defect1: unable to print in pdf.

Defect2: unable to print in pdf and xml.



# 1 软件缺陷的去噪

## □计算以下两个BugReport的相似度

Defect1: unable to print in pdf.

Defect2: unable to print in pdf and xml.

### N-gram法:

- 首先以字母为单位, 用3-gram为例获取两个缺陷如下的字符串序列:

Defect1: {unable,unable to,to print,print in,in pdf,pdf};

Defect2: {unable,unable to,to print,print in,in pdf,pdf and,  
and xml,xml}。

- 其次计算字符串序列中的项目总个数以及两个文本中相同的子项目数:

1. totalItems = Defect1的项目个数+ Defect2的项目个数= 6 + 8 = 14;

2. 相同的子项目数count = 5 × 2;

- 最后计算N-gram相似度:

$$Sim - NG = (5 \times 2) / 14 = 0.7143$$

# 1 软件缺陷的去噪

## □计算以下两个BugReport的相似度

Defect1: unable to print in pdf.

Defect2: unable to print in pdf and xml.

### VSM法:

- 首先统计索引词的各种出现频率，上述例子中共两个文档、七个索引词 (unable, to, print, in, pdf, and, xml)，定义并统计如下内容：
  - 1.freq[i][j]:表示第i个单词在第j个文档中出现的次数；
  - 2.maxfreq[j]:表示第j个文档中出现次数最多的单词的出现次数；
  - 3.termWeight[i][j]:第i个单词在第j个文档中的权重；

# 1 软件缺陷的去噪

## □计算以下两个BugReport的相似度

Defect1: unable to print in pdf.

Defect2: unable to print in pdf and xml.

- 其次计算每个单词在每个文档中的权重，以termWeight[1][0]为例说明，即第2个索引词 (to) 在第1个文档 (Defect1) 中的权重。其余类似。

$$1. \text{tf}[1][0] = \text{freq}[1][0] / \text{maxfreq}[0]$$

= 第2个词在第1个文档中的次数 / 第1个文档中单个词的最高次数

$$= 1/1$$

$$= 1$$

$$2. \text{idf}[1] = \log(N/n_1 + \text{offset})$$

= log (文档总个数 / 第2个词出现的文档总个数) + offset

$$= \log(2/2 + 0.1) = \log(1.1) = \ln(1.1) = 0.09531$$

$$3. \text{termWeight}[1][0] = \text{tf}[1][0] \times \text{idf}[1]$$

$$= 1 \times 0.09531 = 0.09531$$



# 1 软件缺陷的去噪

## □ 计算以下两个BugReport的相似度

Defect1: unable to print in pdf.

Defect2: unable to print in pdf and xml.

- 然后构造缺陷文档向量，结果如下：

$$\begin{aligned}\text{Defect1} &= \{\text{termWeight}[0][0], \text{termWeight}[1][0], \dots, \text{termWeight}[6][0]\} \\ &= \{0.09531, 0.09531, 0.09531, 0.09531, 0.09531, 0, 0\}\end{aligned}$$

$$\begin{aligned}\text{Defect2} &= \{\text{termWeight}[0][1], \text{termWeight}[1][1], \dots, \text{termWeight}[6][1]\} \\ &= \{0.09531, 0.09531, 0.09531, 0.09531, 0.09531, 0.7419, 0.7419\}\end{aligned}$$

- 最后计算Cos相似度：

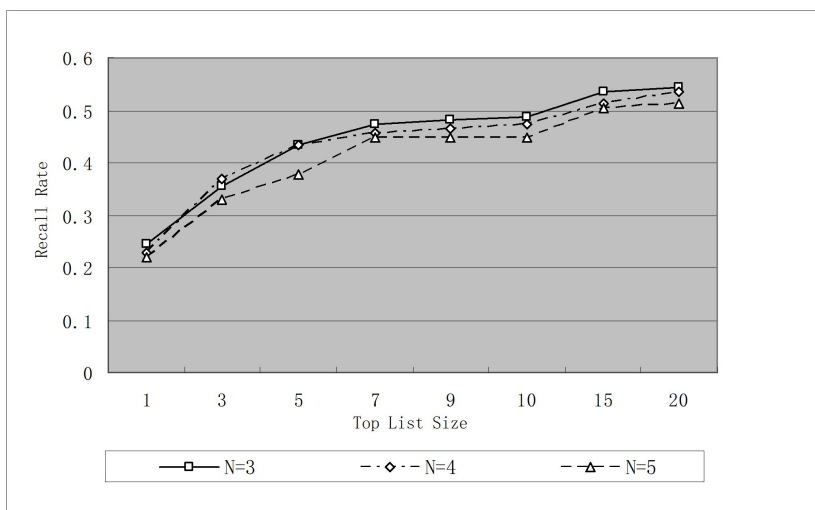
$$\text{Sim} - \text{Cos} = \frac{0.09531 \times 0.09531 + \dots + 0 \times 0.7419 + 0 \times 0.7419}{\sqrt{0.09531^2 + \dots + 0^2} \times \sqrt{0.09531^2 + \dots + 0.7419^2}} = 0.1991$$

# 1 软件缺陷的去噪

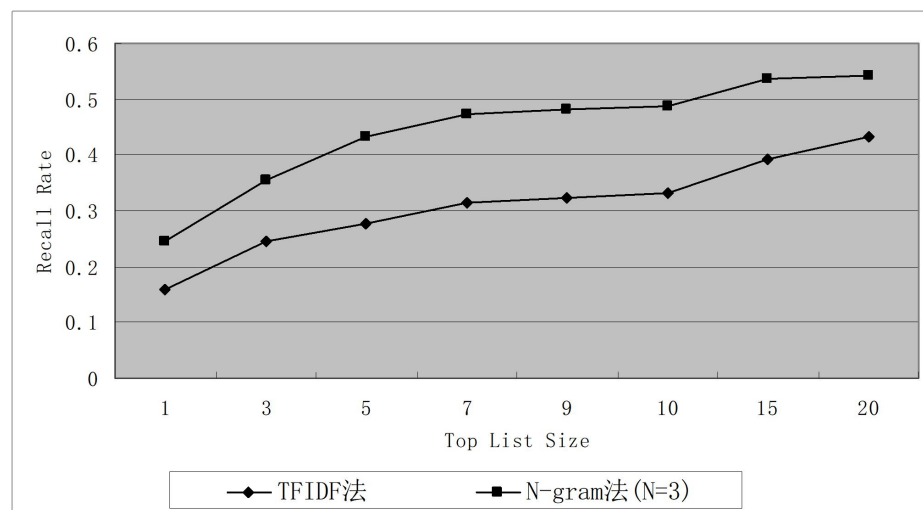
## ◆ 实验与分析 - 基于真实数据集的实验

实验数据：4503条开源软件Firefox的缺陷数据

实验方法：全句法， $N=3/4/5$ ，仅利用缺陷的概述文本信息



N-gram的参数N为3时查全率最高



N-gram法的查全率  
优于VSM法

1. 软件缺陷数据预处理

2. 软件缺陷分类

3. 软件缺陷预测

4. 软件缺陷智能分析



# 2 软件缺陷分类

## □ 预测目标

- 缺陷类型（包含多分类）
- 缺陷修复者
- 缺陷严重程度
- 预测某文件是否有缺陷
- 预测某次提交修改是否有缺陷Just-in-time

## □ 数据来源

- 缺陷报告
- 项目、产品、过程等度量数据

## □ 预测技术

- 缺陷文本挖掘
- 非文本数据挖掘

## 2 软件缺陷分类-文本分类

### □ 基于文本分类的缺陷类型自动分类

- BugClassifier工具

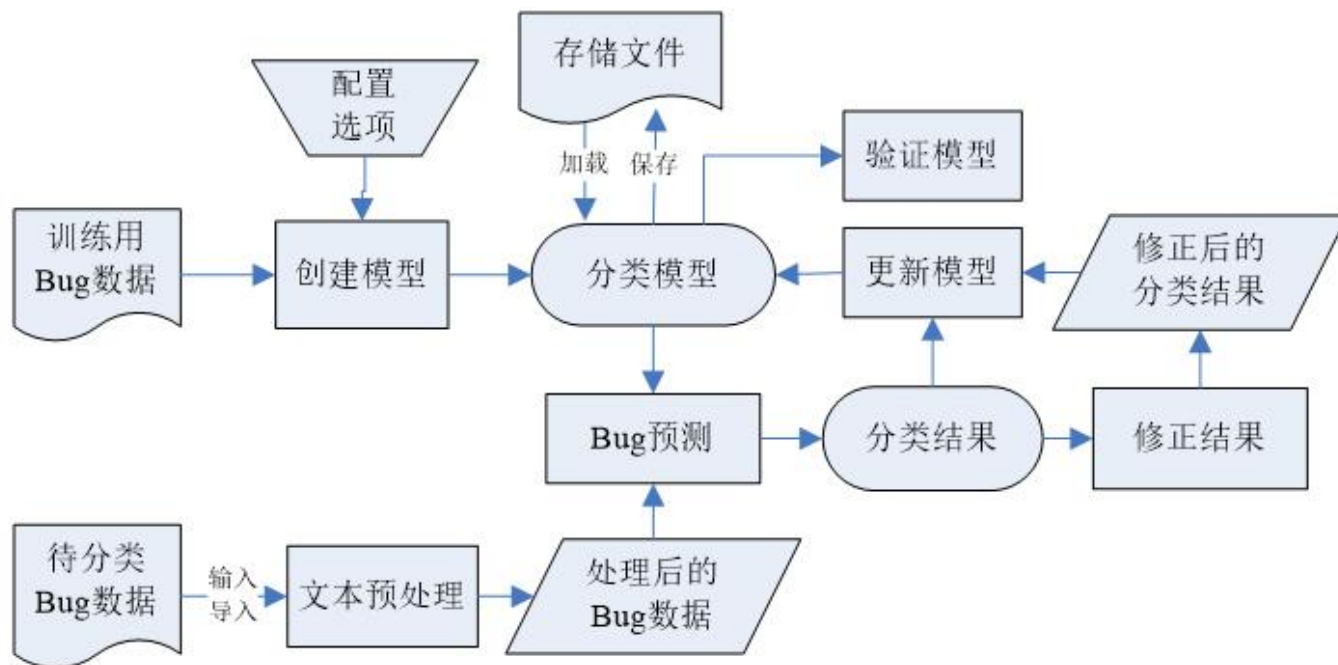
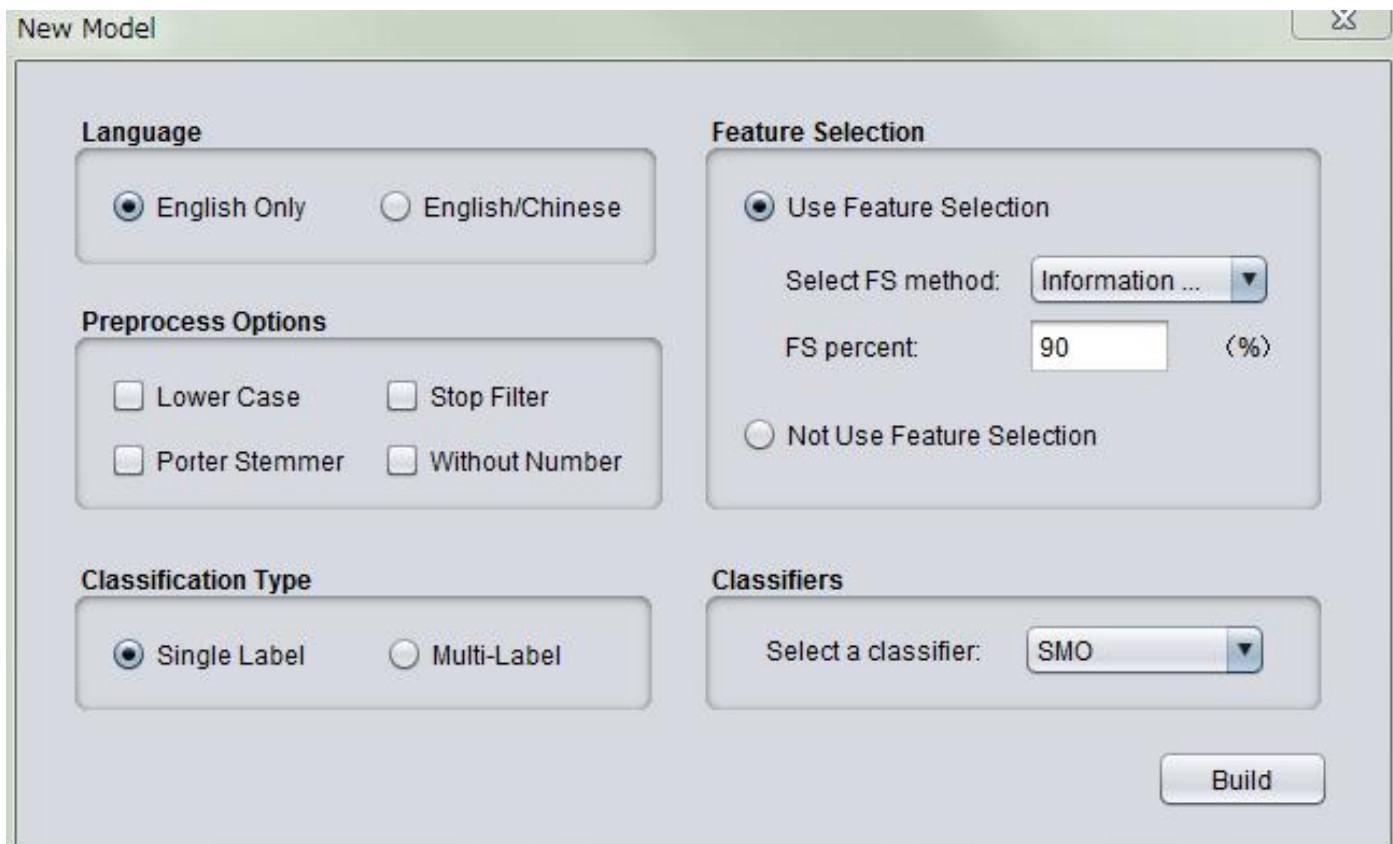


图 5.1-2 操作流程图

## 2 软件缺陷分类-文本分类

### □ 基于文本分类的缺陷类型自动分类

- BugClassifier工具

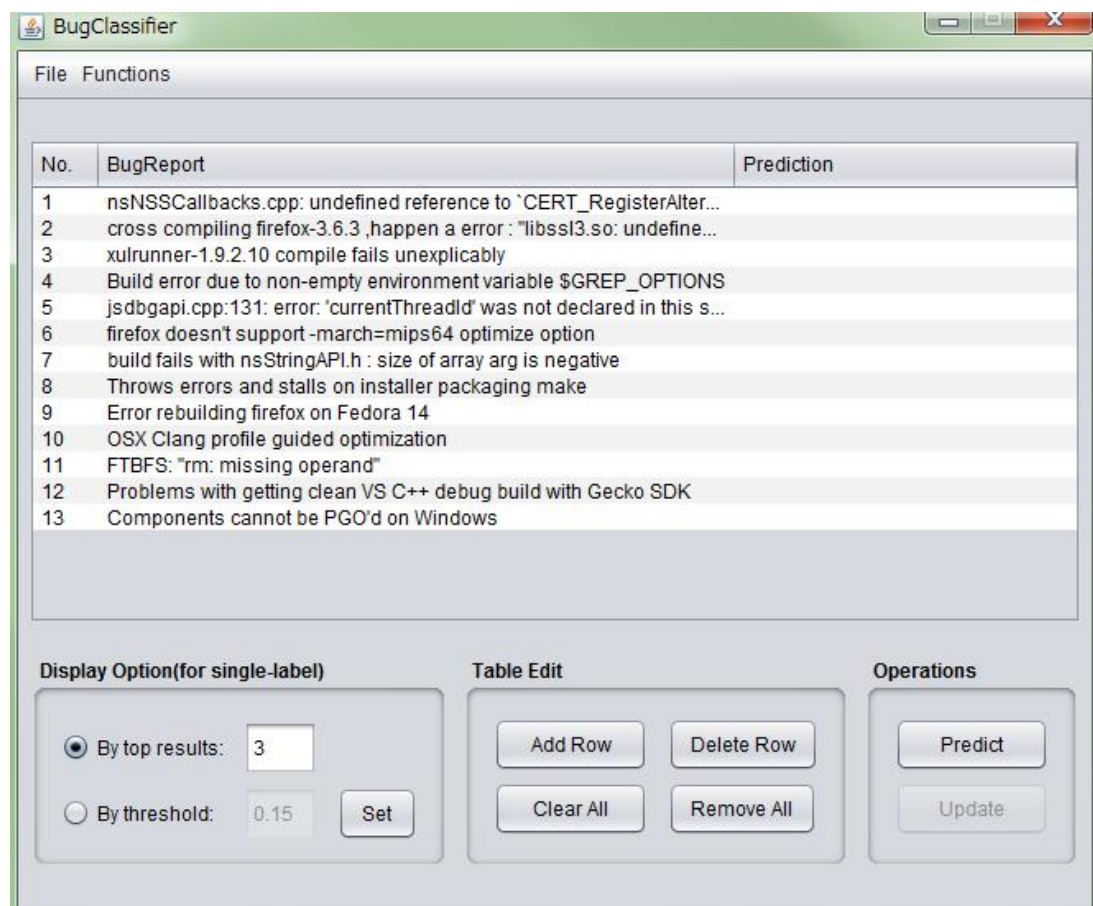


The image shows the 'New Model' dialog box of the BugClassifier tool. It contains several configuration sections:

- Language:** Radio buttons for 'English Only' (selected) and 'English/Chinese'.
- Preprocess Options:** Checkboxes for 'Lower Case', 'Stop Filter', 'Porter Stemmer', and 'Without Number'.
- Feature Selection:** Radio buttons for 'Use Feature Selection' (selected) and 'Not Use Feature Selection'. Below 'Use Feature Selection' is a 'Select FS method:' dropdown menu (showing 'Information ...') and an 'FS percent:' input field (showing '90') with a '(%)' suffix.
- Classification Type:** Radio buttons for 'Single Label' (selected) and 'Multi-Label'.
- Classifiers:** A 'Select a classifier:' dropdown menu (showing 'SMO').
- Build:** A button at the bottom right.

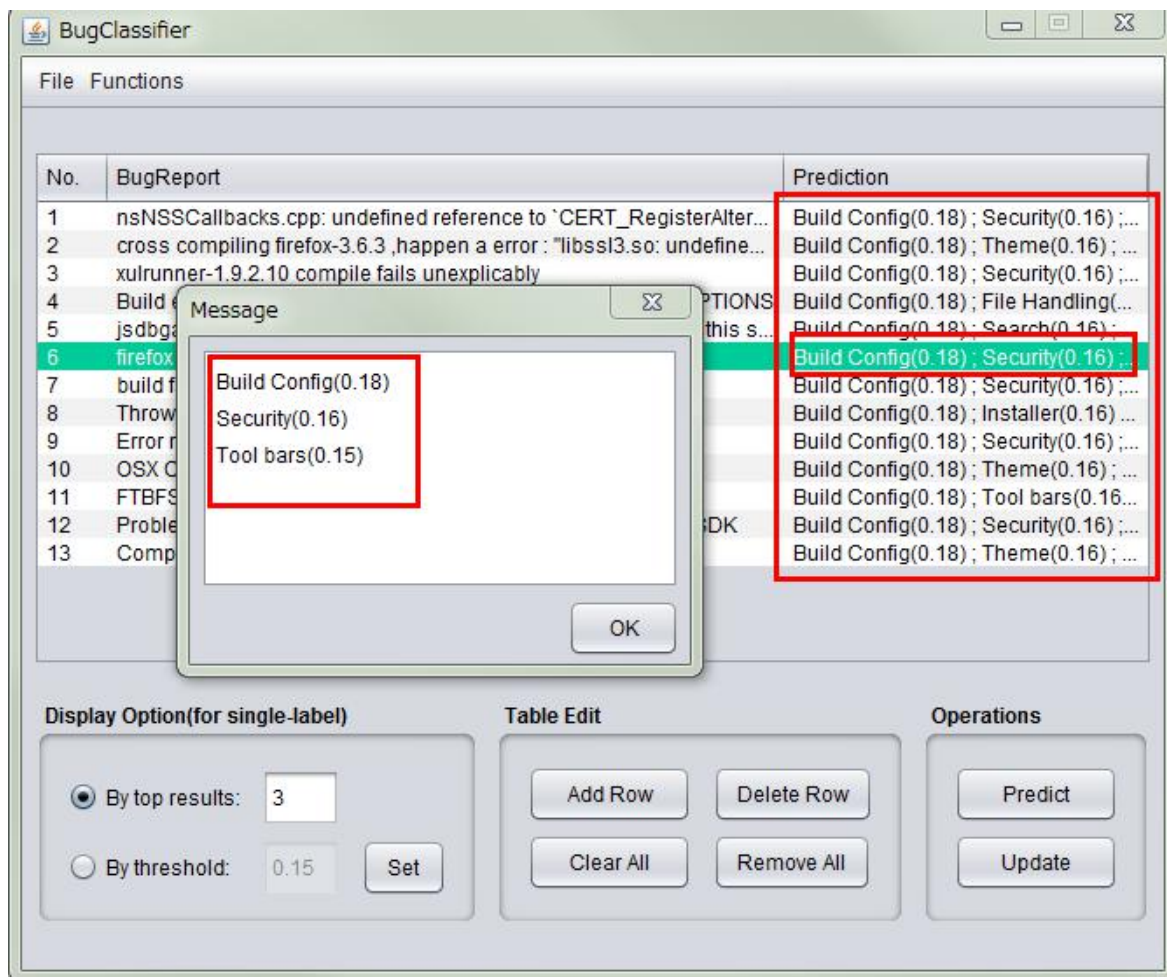
## 2 软件缺陷分类-文本分类

- 基于文本分类的缺陷类型自动分类
  - BugClassifier工具



## 2 软件缺陷分类-文本分类

- 基于文本分类的缺陷类型自动分类
  - BugClassifier工具



## 2 软件缺陷分类-文本分类

### □ 基于文本分类的缺陷类型自动分类

- AutoODC: Automated Generation of ODC

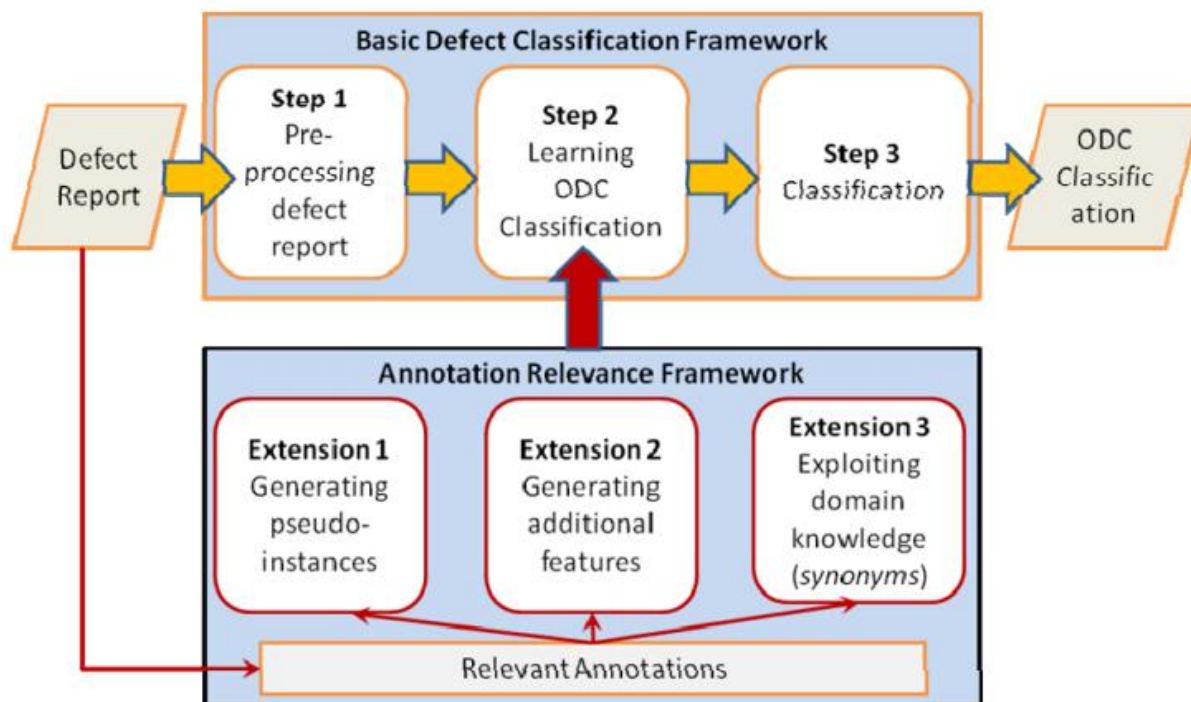


Figure 1. AutoODC Overview.

AutoODC: Automated generation of orthogonal defect classifications, ASE2015



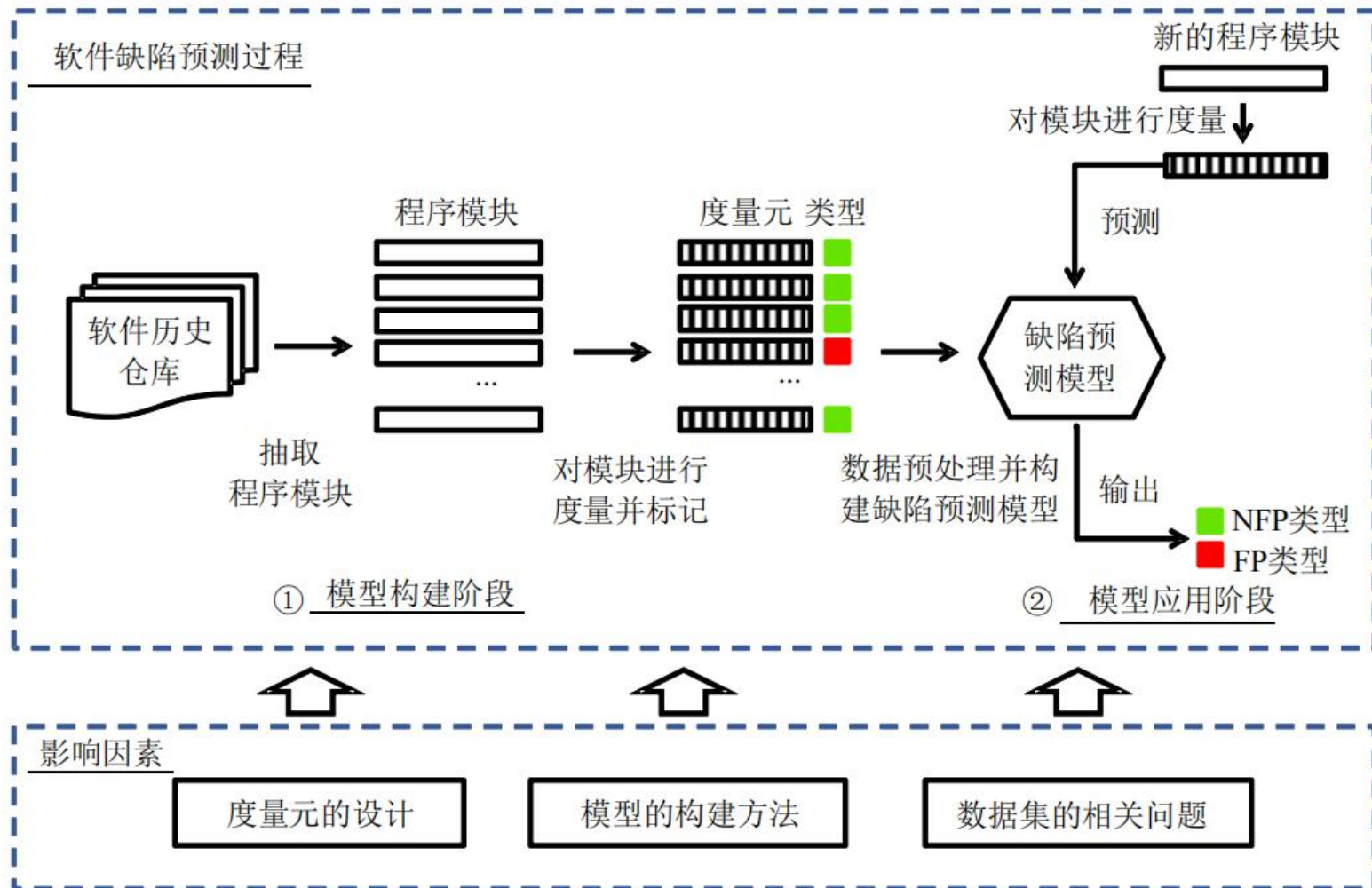
## 2 软件缺陷分类-缺陷倾向性预测

### □模块缺陷倾向性预测（有/无缺陷分类）

- 预测用的数据筛选: 建立预测模型所用到的基础数据来源（公开[NASA MDP]、私有、半公开、未知）
- 预测用的数据度量: 确定数据由哪些属性构成（产品度量、过程度量、项目度量）
- 预测的方法模型: 预测的方法和模型（统计学、机器学习、统计学+专家知识、统计学+数据挖掘等）
- 预测的对象: 某个模块或者某次提交是否有缺陷

## 2 软件缺陷分类-缺陷倾向性预测

### □ 模块缺陷倾向性预测（有/无缺陷分类）





## 2 软件缺陷分类-缺陷倾向性预测

### □模块缺陷倾向性预测（有/无缺陷分类）

```
1 @RELATION ant-1.3-ckmetrics
2
3 @ATTRIBUTE wmc REAL
4 @ATTRIBUTE dit REAL
5 @ATTRIBUTE rfc REAL
6 @ATTRIBUTE noc REAL
7 @ATTRIBUTE cbo REAL
8 @ATTRIBUTE lcom REAL
9 @ATTRIBUTE loc REAL
10 @ATTRIBUTE isBug {YES,NO}
11
12 @DATA
13 11,4,42,2,14,29,395,NO
14 14,1,32,1,8,49,257,YES
15 3,2,9,0,1,0,58,NO
16 12,3,37,0,12,32,310,NO
17 6,3,21,0,4,1,136,NO
18 5,1,15,0,3,0,137,NO
19 4,4,32,0,6,6,325,NO
20 16,3,40,0,3,84,604,NO
21 4,5,21,0,5,0,87,NO
22 17,3,55,0,10,76,461,YES
23 9,3,29,0,9,0,168,NO
24 3,3,14,0,3,1,84,YES
25 10,3,41,0,5,21,360,NO
26 17,2,64,0,9,76,713,YES
27 5,1,11,5,12,8,59,NO
28 10,1,49,0,9,1,382,NO
29 22,4,72,0,16,165,675,YES
30 5,1,13,0,4,0,65,NO
```

#### ➤ 研究热点:

- 有监督、半监督与无监督预测
- 项目内预测与跨项目预测（迁移学习）
- 不均衡数据
- 代价敏感
- Just-in-time

## 2 软件缺陷分类-缺陷倾向性预测

### □模块缺陷倾向性预测 — 预测性能度量

Table 2: Confusion Matrix

	Observed defective	Observed defect free
Predicted defective	True Positive (TP)	False Positive (FP)
Predicted defect free	False Negative (FN)	True Negative (TN)

1. FPR (False Positive Rate) :  $\frac{FP}{TN+FP}$ .
2. FNR (False Negative Rate) :  $\frac{FN}{TP+FN}$
3. ER (Error Rate):  $\frac{FP+FN}{TP+TN+FP+FN}$
4. Recall:  $\frac{TP}{TP+FN}$
5. Accuracy:  $\frac{TP+TN}{TP+TN+FP+FN}$
6. Precision:  $\frac{TP}{TP+FP}$
7. FMeasure (F1):  $\frac{2 \times Recall \times Precision}{Recall + Precision}$
8. MCC:  $\frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$  [8]
9. AUC: Area Under the Curve (AUC) ROC chart [9].
10. Popt/ACC: Effort-aware prediction performance [10].
11. MeanAIC: The mean of Akaike's Information Criterion (AIC) ([P32]).
12. MAE: Mean absolute error between prediction and observation ([P10]).
13. G-Mean:  $\sqrt{\frac{TP}{TP+FN} * \frac{TN}{TN+FP}}$  ([P47]).

思考:

1. 不同度量的优缺点 ?
2. 如果进行不同算法的预测性能比较 ?

## 2 软件缺陷分类-缺陷倾向性预测

### □ 模块缺陷倾向性预测（有/无缺陷分类）

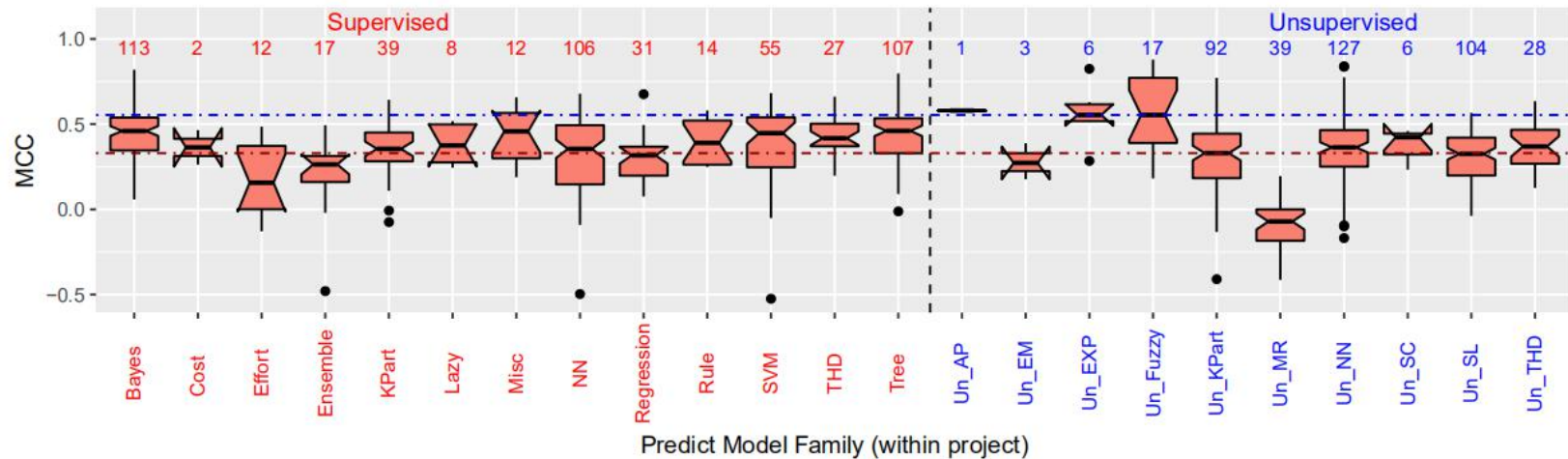


Figure 5: Defect prediction models in within-project prediction. NB the numbers indicate the count of experiments (sample size).

## 2 软件缺陷分类-缺陷倾向性预测

### □ 根据原始缺陷预测实验的数据进行混淆矩阵的反算

表 1 使用三种性能度量反算混淆矩阵组合

	FPR	r	a	p	F1	混淆矩阵是否可算
1	✓	✓	✓			可算
2	✓	✓		✓		可算
3	✓	✓			✓	可算
4	✓		✓	✓		可算
5	✓		✓		✓	可算
6	✓			✓	✓	可算

1. 已知 FPR, r, a, 求解混淆矩阵的各个值。

$$TP = \frac{r(1-a-FPR)}{1-r-FPR}$$

$$FP = \frac{FPR(a-r)}{1-r-FPR}$$

$$TN = \frac{(1-FPR)(a-r)}{1-r-FPR}$$

$$FN = \frac{(1-r)(1-a-FPR)}{1-r-FPR}$$

2. 已知 FPR, r, d, 求解混淆矩阵的各个值。

$$FP = FPR * (1-d)$$

$$TN = (1-d) * (1-FPR)$$

$$FN = d * (1-r)$$

$$TP = 1 - FP - TN - FN$$

**可选作业：度量转换工具实现**

1. 软件缺陷数据预处理

2. 软件缺陷分类

3. 软件缺陷预测

4. 软件缺陷智能分析

## □ 缺陷预测的内容

软件缺陷相关的预测内容是软件开发测试计划的重要组成部分

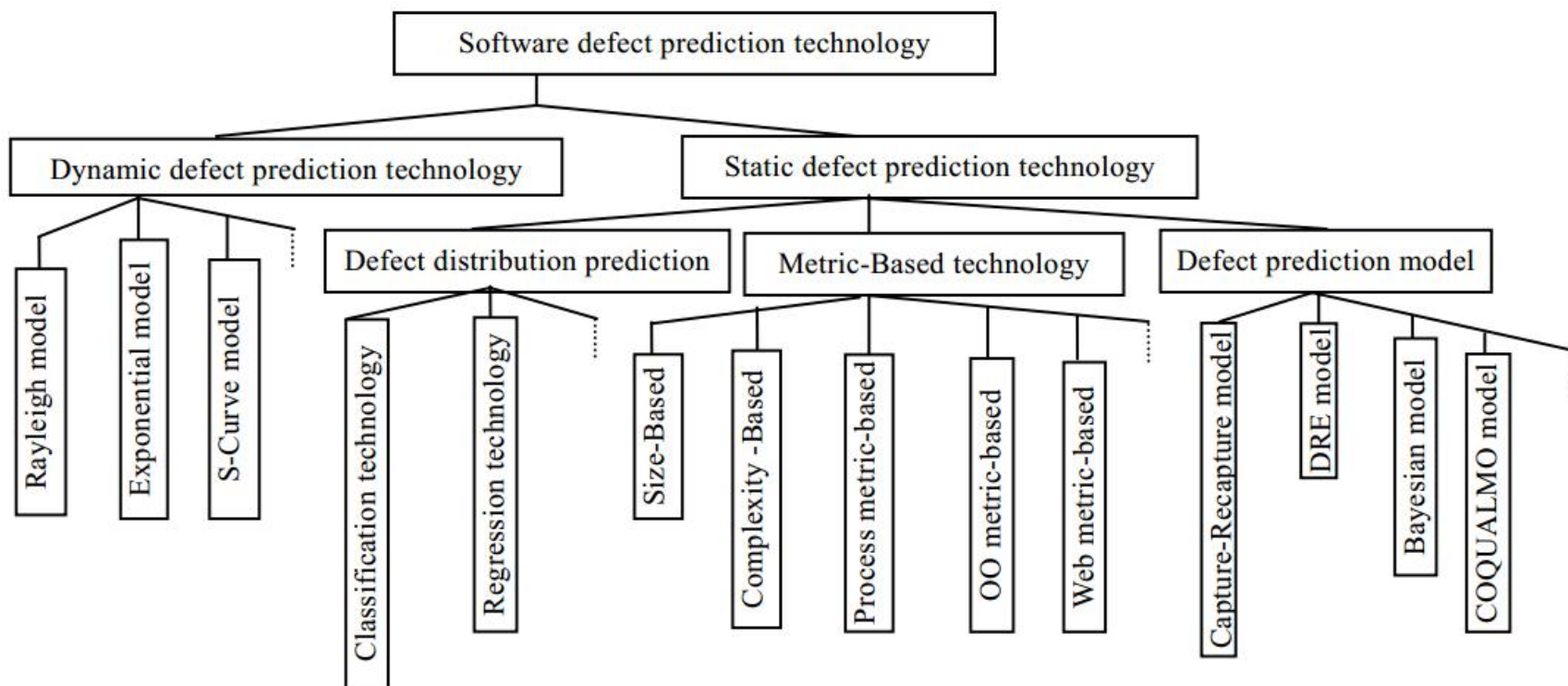
分类与预测的区别：类型预测 / 数值预测

### ➤ 预测的对象: 预测的具体目标对象

- 修复缺陷的时间
- 缺陷密度
- 缺陷分布



## □ 缺陷预测方法



# 3 缺陷预测技术

预测的部分方法介绍:

1. Akiyama: 线性回归

$$D=4.86+0.018L;$$

2. 基于软件复杂度的缺陷预测技术

$CC(1\sim10), 5\%; CC(20\sim30), 20\%; CC(>50), 40\%; CC(\text{接近 } 100), 60\%;$

含义: CC表示圈复杂度, 当CC在1-10时, 修复缺陷可能引入新缺陷的概率是5%。

3. 基于分布和数据挖掘技术。



## □ 缺陷预测 - DRE模型

Table 3 DRE model used in practice

表 3 DRE 模型实例数据

Defect removal step	Defect injection phase				Total
	Requirements	High-Level design	Low-Level design	Coding	
Requirement analysis and review	13				13
High-Level design inspections	2	8			10
Low-Level design inspections	2	3	7		12
Code inspections and Testing	2	2	1	18	23
Customer detected	1	1	1	2	5
Total	20	4	9	20	63

缺陷移除率 =  $\frac{\text{当前阶段工作实际发现的缺陷数量}}{\text{当前阶段应该发现的缺陷数量的比值}}$

整个系统的缺陷移除效率= $(13+10+12+23)/63=92\%$ .

各个阶段的缺陷移除效率:需求阶段  $DRE=13/(13+2+2+2+1)=0.65$ ;概要设计阶段  $DRE=10/((2+2+2+1)+(8+3+2+1))=0.33$ ;详细设计阶段  $DRE=12/((2+2+1)+(3+2+1)+(7+1+1))=0.6$ ;代码评审与测试阶段  $DRE=23/((2+1)+(2+1)+(1+1)+(18+2))=0.82$ ;用户发现  $DRE=5/(1+1+1+2)=1.00$ .

设计移除缺陷的效率最低,测试在缺陷移除过程中最有效。

# 3 缺陷预测-数据挖掘

## □ 缺陷自动预测 - 贝叶斯网络模型

表 6.3 拥有14条实例的天气数据集

outlook	temperature	humidity	windy	play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

## □ 缺陷自动预测 - 贝叶斯网络模型

定义 6.1 ( 贝叶斯定理 ): 设 $X$ 是类标号未知的样本数据。设 $H$ 为某种假设, 如数据样本 $X$ 属于某特定的类 $C$ 。对于类别预测问题, 希望确定 $P(H|X)$ , 即给定观测数据样本 $X$ , 假定 $H$ 成立的概率, 根据贝叶斯定理计算:

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)} \quad (6-1)$$

其中 $P(H)$ 是先验概率,  $P(X|H)$ 表示假设 $H$ 成立的条件下, 观察到 $X$ 的概率。 $P(H|X)$ 是后验概率, 或称条件 $X$ 下 $H$ 的后验概率。

定义 6.2 ( 贝叶斯网络中的联合概率 ): 设 $X = (x_1, x_2, \dots, x_n)$ 是描述变量或者属性 $Y_1, Y_2, \dots, Y_n$ 的一个数据元组。注意, 给定变量的双亲, 每个变量都有条件地独立于网络图中的它的非后代。这使得网络中结点值组合的联合概率可以用公式 (6-2) 表示。其中,  $P(x_1, x_2, \dots, x_n)$ 是 $X$ 的值的特定组合的概率,  $Parents(Y_i)$ 是 $x_i$ 的双亲结点集,  $P(x_i|Parents(Y_i))$ 的值对应于 $Y_i$ 的CPT表。

$$P(x_1, x_2, \dots, x_n) = \prod_{i=1}^n P(x_i|Parents(Y_i)) \quad (6-2)$$

## □ 缺陷自动预测 - 贝叶斯网络模型

在实际计算中，为了避免出现概率值为0的情况，每个结点自身的概率按照m-估计的公式 (6-3) 计算：

$$\frac{n_c + mp}{n + m} \quad (6-3)$$

其中对于 $P(H|X)$ ， $n$ 表示满足条件 $X$ 的样本数目， $n_c$ 表示在 $n$ 个满足条件 $X$ 的样本中符合假设 $H$ 的样本数目。 $P$ 是先验概率， $m$ 是一个等效样本大小的常量，此处 $m=1$ 。

贝叶斯网络 = 有向图 + 条件概率表CPT



# 3 缺陷预测-数据挖掘

## ❑ 缺陷自动预测 - 贝叶斯网络模型

### 1) 构建贝叶斯网络:

根据人们对阴晴、刮风、湿度、温度以及玩之间关系的经验知识，构建贝叶斯网络。

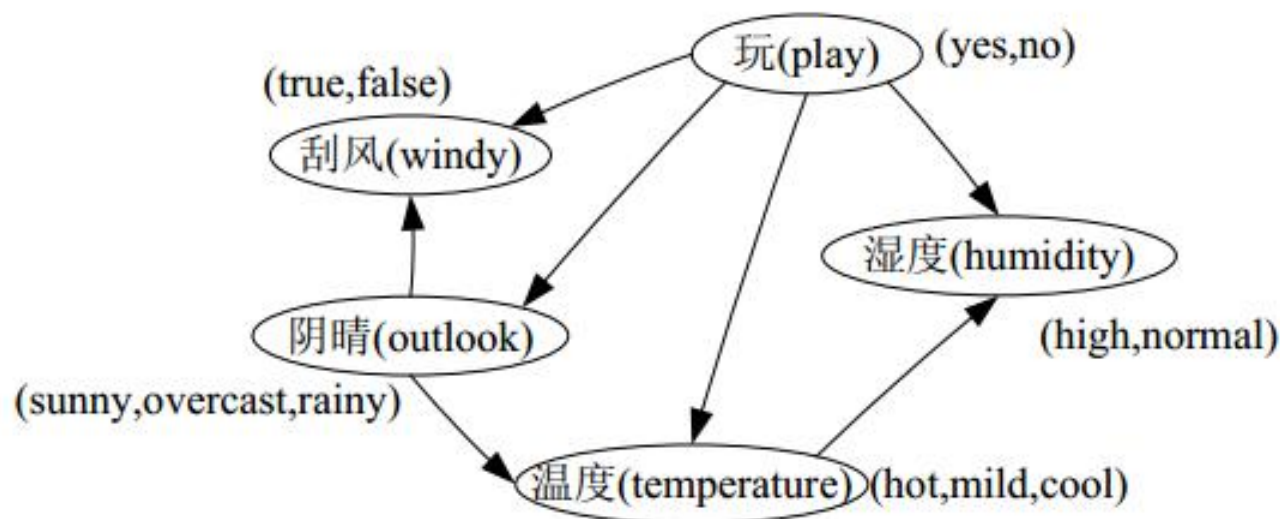


图 6.3 天气数据的一个简单贝叶斯网络结构图

## □ 缺陷自动预测 - 贝叶斯网络模型

### 2) 根据已有的数据集计算每个结点的条件概率表CPT

- 根据公式 (6-3) 计算结点“玩”的CPT。

$m=1$ ,  $p$ 是“玩”为yes和no的平均概率, 由于只有2个值, 所以 $p=0.5$ 。该结点的CPT表中信息如下:

$$P(\text{play} = \text{yes}) = \frac{9+1 \times 0.5}{14+1} = 0.633。 (\text{总样本} n=14; \text{play}=\text{yes} \text{的样本} n_c = 9。)$$

$$P(\text{play} = \text{no}) = \frac{5+1 \times 0.5}{14+1} = 0.367。 (\text{总样本} n=14; \text{play}=\text{yes} \text{的样本} n_c = 5。)$$

- 根据公式 (6-3) 计算结点“湿度 (humidity)”的CPT。

$m=1$ ,  $p$ 是“湿度”为high和normal的平均概率, 由于只有2个值, 所以 $p=0.5$ 。该结点第一条数据的计算过程如下, 其余类似。

$$P(\text{humidity} = \text{high} | \text{play} = \text{yes}, \text{temperature} = \text{hot}) = \frac{1+1 \times 0.5}{2+1} = 0.5。$$

(满足 $\text{play}=\text{yes}$ ,  $\text{temperature}=\text{hot}$ 的样本 $n=2$ ; 在这2个样本中,  $\text{play}=\text{yes}$ 的样本有1个, 即 $n_c = 1$ 。)

- 其余的依次按照上述方法计算, 最终所有结点的计算结果如图6.4所示。

# 3 缺陷预测-数据挖掘

## □ 缺陷自动预测 - 贝叶斯网络模型

3) 根据建立好的贝叶斯网络模型进行预测： 阴晴=rainy, 、  
温度=cool, 湿度=high, 刮风=true的天气下, 是否出去玩

$$P(\text{play} = \text{no}, \text{outlook} = \text{rainy}, \text{temperature} = \text{cool}, \text{humidity} = \text{high}, \text{windy} = \text{true})$$

$$\begin{aligned} &= P(\text{play} = \text{no}) \times P(\text{outlook} = \text{rainy} | \text{play} = \text{no}) \\ &\quad \times P(\text{temperature} = \text{cool} | \text{play} = \text{no}, \text{outlook} = \text{rainy}) \\ &\quad \times P(\text{humidity} = \text{high} | \text{play} = \text{no}, \text{temperature} = \text{cool}) \\ &\quad \times P(\text{windy} = \text{true} | \text{play} = \text{no}, \text{outlook} = \text{rainy}) \\ &= 0.367 \times 0.385 \times 0.429 \times 0.25 \times 0.167 = 0.0025 \end{aligned}$$

$$P(\text{play} = \text{yes}, \text{outlook} = \text{rainy}, \text{temperature} = \text{cool}, \text{humidity} = \text{high}, \text{windy} = \text{true})$$

$$\begin{aligned} &= P(\text{play} = \text{yes}) \times P(\text{outlook} = \text{rainy} | \text{play} = \text{yes}) \\ &\quad \times P(\text{temperature} = \text{cool} | \text{play} = \text{yes}, \text{outlook} = \text{rainy}) \\ &\quad \times P(\text{humidity} = \text{high} | \text{play} = \text{yes}, \text{temperature} = \text{cool}) \\ &\quad \times P(\text{windy} = \text{true} | \text{play} = \text{yes}, \text{outlook} = \text{rainy}) \\ &= 0.633 \times 0.333 \times 0.333 \times 0.125 \times 0.875 = 0.0077 \end{aligned}$$



# 3 缺陷预测-数据挖掘

## □ 缺陷自动预测 - 贝叶斯网络模型

3) 根据建立好的贝叶斯网络模型进行预测：阴晴=rainy, 温度=cool, 湿度=high, 刮风=true的天气下，是否出去玩

因此：

$$P(\text{play} = \text{no}) = 0.0025 / (0.0025 + 0.0077) = 0.245$$

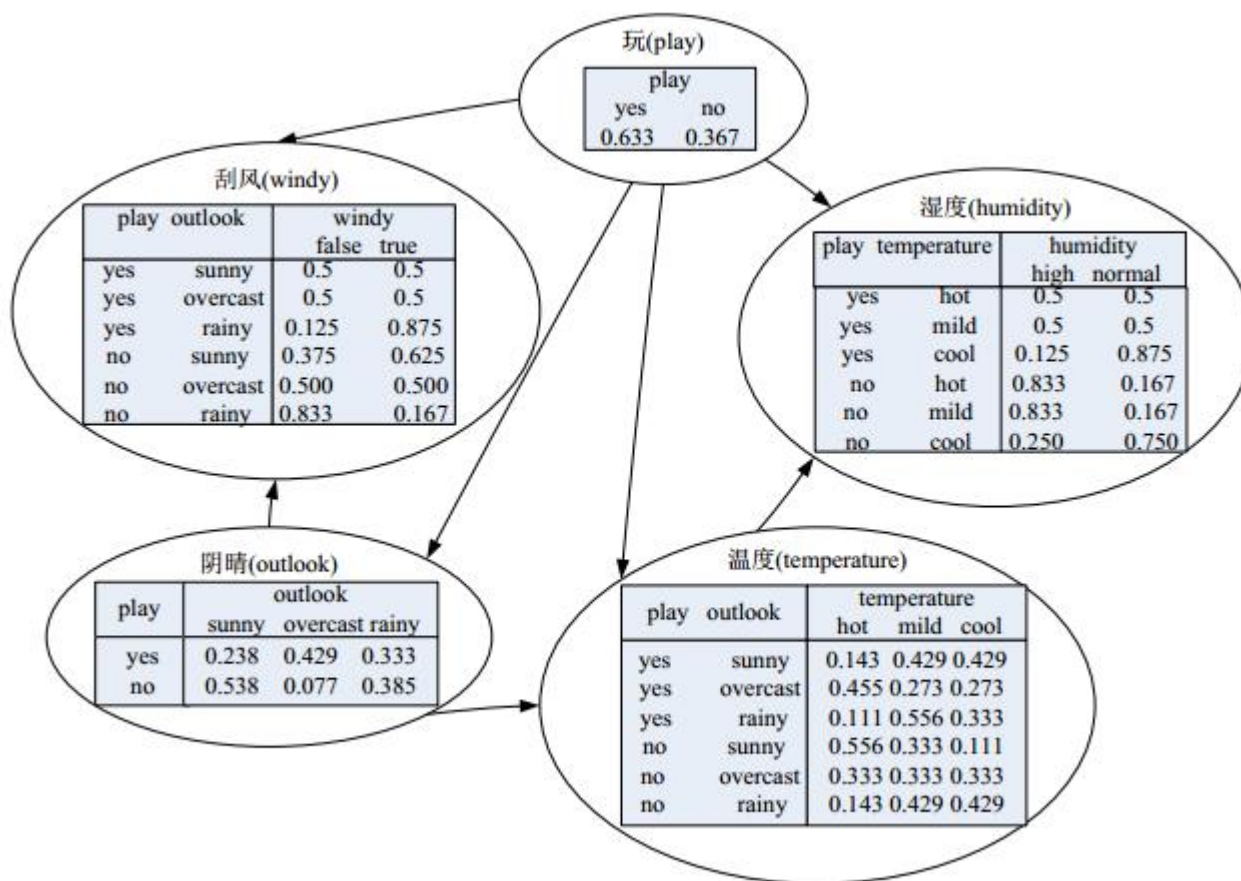
$$P(\text{play} = \text{yes}) = 0.0077 / (0.0025 + 0.0077) = 0.755$$

所以，最终不玩的概率是0.245，玩的概率是0.755。

# 3 缺陷预测-数据挖掘

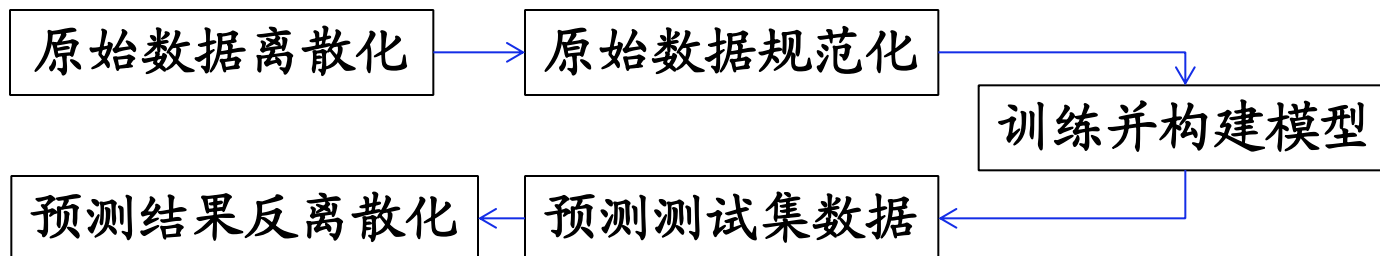
## ❑ 缺陷自动预测 - 贝叶斯网络模型

2) 根据已有的数据集计算每个结点的条件概率表CPT



### 3 缺陷预测-数据挖掘

#### ❑ 缺陷自动预测 - 贝叶斯网络模型



### 3 缺陷预测-数据挖掘

#### □ 缺陷自动预测 - 贝叶斯网络模型

规模	稳定程度	...	测试用例	管理水平	管理经验	...	测试方法熟悉度	测试方法熟悉度	测试工时	缺陷个数	缺陷密度
32.5	M	...	1163	M	0.1	...	M	M	736	16	0.49
1.3	H	...	89	M	0.2	...	M	L	223	0	0
8.88	M	...	561	M	0.3	...	M	M	490	5	0.56
17.1	H	...	727	H	0.4	...	M	H	598	6	0.35
...	...	...	...	...	...	...	...	...	...	...	...
3	M	...	178	M	1.65	...	M	M	341	2	0.67
4.5	H	...	272	M	3.1	...	M	M	223	1	0.22

### 3 缺陷预测-数据挖掘

#### □ 缺陷自动预测 - 基于层叠泛化的预测模型

层叠泛化思想

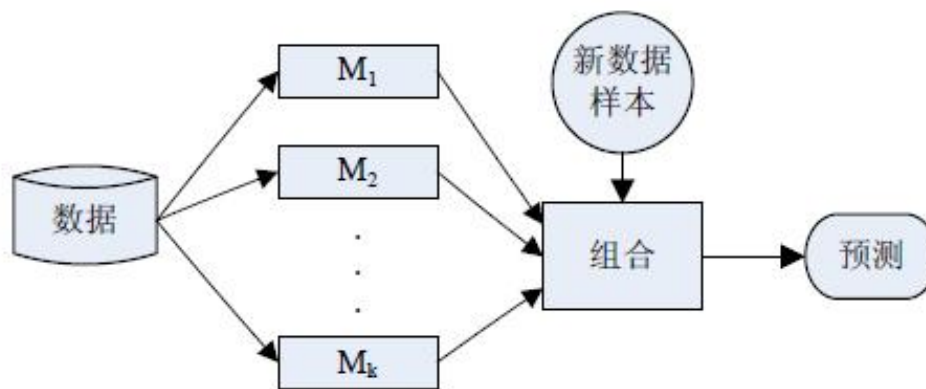


图 6-1 模型组合示意图

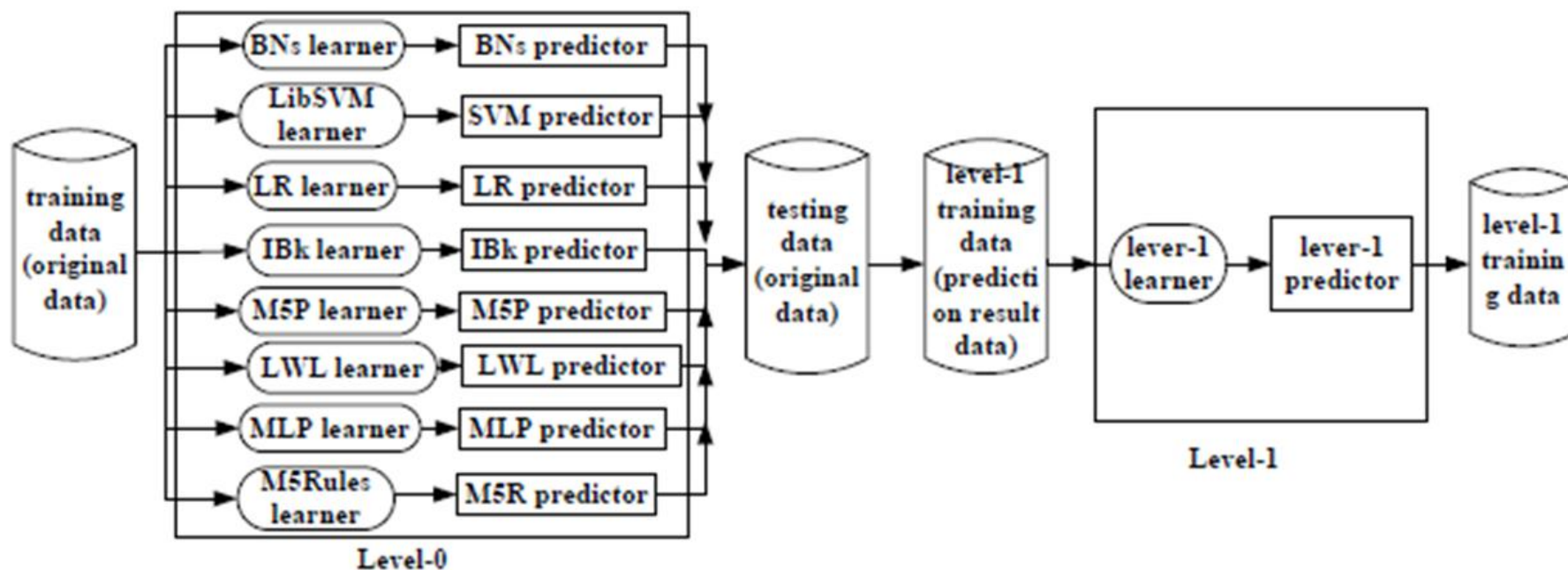
层叠泛化的方法中，将要组合的多个模型分为0层模型 (level-0 model) 和1层模型 (level-1 models)。通常情况下，整个训练预测的过程分为两步进行：

1. 用0层模型进行训练和预测。针对原始数据，利用n种0层模型进行训练，得到n种模型的预测结果。
2. 用1层模型进行训练和预测。针对0层模型得到的n个预测结果，利用1层模型进行训练，得到最终的预测结果。



### 3 缺陷预测-数据挖掘

#### □ 缺陷自动预测 - 基于层叠泛化的预测模型



1. 软件缺陷数据预处理

2. 软件缺陷分类

3. 软件缺陷预测

4. 软件缺陷智能分析



## □ 基于缺陷类型(单个属性)的频繁缺陷挖掘

方法: Apriori算法     分析数据: 已分过类的1860条Firefox缺陷数据

挖掘结果: (1) 最频繁出现的10个条件类型(数据或者操作)。

- 不存在值/对象 [8.61%]
- 边界值 [8.44%]
- 特殊字符 [6.33%]
- 空值 [5.62%]
- 大/长/深值 [5.45%]
- 不匹配的类型/格式 [4.39%]
- 相同/相等值 [3.69%]
- 移动/拖放 [3.69%]
- 重复操作 [2.99%]
- 并发操作 [2.46%]

可以指导测试人员如何设计测试用例的条件会更容易发现缺陷。

## □ 基于多个分类属性的频繁缺陷挖掘

挖掘结果: (2) 结合“缺陷类型”和“影响度”两个属性, 挖掘对用户影响度为“高”或者“中”, 并且最常出现的结果类型频繁项集。

该结果可以让开发人员和测试人员清楚地意识到这些问题, 从而加强这些方面的测试。

- 未捕捉或者未处理的异常 [8.18%]
- 画面内容异常(无响应/不能显示/错乱/抖动等) [7.06%]
- 控件状态异常(有效无效/可编辑与否/选中与否/显示与否) [4.09%]
- 控件功能异常(不工作或者功能错误) [3.07%]
- 程序崩溃 [2.66%]
- 数据内容读写错误 [2.35%]
- 数据自身/关联数据未被修改或者无法被修改 [2.35%]
- 数据自身/关联数据未被删除或者无法被删除 [1.84%]
- 无法终止/死循环 [1.74%]
- 数据自身/关联数据未被增加或者无法增加 [1.64%]

# 4 软件缺陷智能分析

## □ 关联规则挖掘

关联规则：关联规则 (Association Rule) 是形如  $A \Rightarrow B$  的蕴含式，其中  $A \subset I$ ,  $B \subset I$ , 且  $A \cap B = \emptyset$ ,  $I$  表示一个项集。A 为前件，B 为后件。

关联规则支持度:  $supp(A \Rightarrow B) = P(A \cup B)$

关联规则置信度:  $conf(A \Rightarrow B) = P(B|A) = \frac{supp(A \Rightarrow B)}{supp(A)}$

美国沃尔玛连锁店超市的真实案例



## □ 关联规则挖掘

挖掘结果:

### 1. 不同缺陷类型之间的关联规则。

- $C72; C169 \Rightarrow R210$ [85.2%]
- $C90 \Rightarrow R210$ [76.4%]
- $C179 \Rightarrow R210; R222$ [68.5%]

### 2. 功能模块与缺陷类型之间的关联规则。

- $C70 \Rightarrow B.1 \wedge C.3$  [89.4%]。该规则表明当使用“自定义值”(第70号缺陷)测试时, 功能B.1和功能C.3中总是出现错误。
- $C69 \wedge A.1 \Rightarrow R163 \wedge B.3$ [82.2%] 该规则表明当测试者在功能A.1中用“边界值”(第69号缺陷)测试时, 功能B.3中出现“溢出错误”(第163号缺陷) 的概率为82.2%。

## ➤ 热点问题调查



- 2017年以来缺陷分类方面的Paper?
- 2017年以来缺陷预测方面的Paper?

---

**业精于勤，荒于嬉！**

---