基本概念：

1. 软件修复解决方案：可检测现场的软件故障，并通过进行必要的调整来响应，以恢复系统的正常运行。在运行时应用于部署的应用程序，以防止或减轻故障。在同一软件实例上多次出现类似故障可能会多次触发同一修复过程。

2.软件修复解决方案：可检测软件故障，定位可应用修复的位置，并进行必要的调整以修复故障，从而防止同一故障导致的进一步故障。这些调整是在内部生成的，比如在测试和设计时，并在源代码级别部署。

3.定位：故障定位技术：识别错误语句。在这种情况下，修复将把发现的错误语句变成正确的语句。固定定位技术：检测可以观察并消除故障影响的语句。

4.修复：

1. 基于生成和验证的（搜索和暴力）
   1. 基于原子操作（实例插入、删除或修改语句或表达式的单个运算符）
   2. 基于模板：可能会定义复杂的更改模式，这些模式会在多个位置一致地影响正在修复的程序，这很难通过原子更改的随机组合获得。
   3. 基于例子：根据从已用于修复程序的修复示例集中提取的一组修改操作。
2. 语义驱动的：问题Prepair正式编码，或者显式地（例如）编码为公式，其解决方案对应于被修复程序的可能修复，或者隐式地编码为分析过程，其结果是修复。无需针对Prepair进行验证。
   1. 程序合成
3. 修复推荐 ：不尝试生成修复的技术，只是简单地建议对软件进行一些更改以修复故障，它们的输出也可以快速转化为修复。

5.可做的点：

1. 平滑适应度函数的梯度可能会改善自动修复过程
2. 选择变更模型的类型和修复故障时应采用的算法策略。
3. 从其他程序获取语句可能有助于修复更多故障。
4. 更改通常可以从程序中已经存在的代码中进行组装
5. 较大的搜索空间不一定对应更有效的修复过程
6. 在生成合理的修复方案方面，随机比基因编程更有效
7. 有证据表明，程序的复杂性和结构（例如，存在可能提高通用程序修复技术有效性的语句）以及注释（例如，存在功能前置和后置条件）对程序修复技术的成功和可应用的类型有影响。
8. 利用尽可能多的信息源非常重要，包括来自其他应用程序和系统的修复、用户提示、规范和注释。到目前为止，只有现有的修复程序被基于示例的技术所利用

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 序号 | 题目 | 年 | 期刊 | 应用 | 修复对象 | 贡献 | 未来工作 | 可做的点 |
| 1 | DeepFix: Fixing Common C Language Errors by Deep Learning | 2017 | AAAI | 源代码 | 学生错误C程序 | DeepFix的核心是一个具有注意力的多层序列对序列神经网络，可以预测错误的程序位置以及所需的正确语句 | 1. 推广到其他编程语言 2. 能够有效处理更长序列的神经网络结构 |  |
| 2 | VuRLE: Automatic Vulnerability Detection and Repair by Learning from Examples | 2017 | ESORICS（顶） | 源代码 | Java 来自GitHub  安卓、网络、文字处理和多媒体应用 | 1）从易受攻击代码及其相应修复代码的示例中学习转换编辑及其上下文（即，描述编辑位置的代码）；2） 聚类相似的转化编辑；3） 提取编辑模式和上下文模式，为每个集群创建多个修复模板。VuRLE使用上下文模式来检测漏洞，并定制相应的编辑模式来修复漏洞。 | 更多漏洞和用各种编程语言编写的应用程序来评估VuRLE |  |
| 3 | A new word embedding approach to evaluate potential fixes for automated program repair | 2018 | IJCNN  CCF C | 源代码 | 评估修复 | 提出应用单词嵌入模型Word2Vec，基于从已知修复语料库中获得的自然度来改进修复评估过程。Word2Vec根据文档向量和softmax输出层的距离评估补丁。 | 模型学习到令牌序列，生成的字向量还可以用于构建变体，以修复bug或帮助修复bug。 |  |
| 4 | An Empirical Investigation into Learning Bug-Fixing Patches in the Wild via Neural Machine Translation | 2018 | ASE  CCF A | 源代码 | JAVA GitHub | 错误代码和相应的固定代码抽象出来，并使用神经机器翻译技术训练一个能够将错误代码转换为固定版本的编解码器模型 |  |  |
| 5 | Syntax and Sensibility: Using Language Models to Detect and Correct Syntax Errors | 2018 | SANER  CCF B | 源代码  AST | 学生错误C程序 | 一种定位语法错误发生位置的方法，对令牌流进行可能的更改，以修复识别的错误。通过使用经过正确源代码培训的语言模型来查找不合适的标记. LSTM |  |  |
| 6 | Learning to Repair Software Vulnerabilities with Generative Adversarial Networks | 2018 | NIPS | 源代码 | SATE IV是一个包含C/C合成代码示例（函数）的数据集 | 对抗性学习方法，从一个离散的源域映射到另一个目标域，而不需要成对标记的示例或源域和目标域是双射。两个新的生成器损失函数. |  |  |
| 7 | Automated Classification of Overfitting Patches with Statically Extracted Code Features | 2019 | ARXIV | 二进制 | JAVA | 过拟合补丁检测系统ODS来评估APR补丁的正确性。  在抽象语法树（AST）级别提取Java的修补程序和有缺陷的程序代码特性。然后，使用代码特征和补丁正确性标签的监督学习来自动学习概率模型。ODS模型最终可以用于对新的程序修复补丁进行分类。 | 1. 执行中提取特征，例如测试诊断。 2. 补丁排序和bug聚类 |  |
| 8 | AUTOMATIC REPAIR AND TYPE BINDING OF UNDECLARED VARIABLES USING NEURAL NETWORKS | 2019 | ARXIV | 源代码AST | 未声明变量错误 | LSTM ，使用AST的结构语义细节进行训练，其中每个节点表示源代码中出现的结构.  修复未声明的变量错误 | 对不同类型的语法、语义错误和逻辑错误进行自动修复 |  |
| 9 | Learning to Generate Corrective Patches Using Neural Machine Translation | 2019 | ARXIV | 源代码 | Java开源项目Ambari,Camel,Hadoop,JettyWicket | 利用过去的修复来提出当前/未来错误的修复  神经序列到序列模型学习过去修复中相应的预纠正和后纠正代码，Ratchet能够为给定的易出错代码查询生成修复代码 | NMT不能保证涵盖/翻译所有单词 |  |
| 10 | NEURAL PROGRAM REPAIR BY JOINTLY LEARNING TO LOCALIZE AND REPAIR | 2019 | ICML | 源代码 | 变量误用错误 | 提出了在程序令牌序列的顺序编码的多头指针网络，预测给定潜在缺陷位置的修复 variable-misuse bugs。 | 使用（图形模型、指针和图形模型的组合）进行联合定位和修复 |  |
| 11 | SampleFix: Learning to Correct Programs by Sampling Diverse Fixes | 2019 | ARXIV | 源代码 | 学生C程序 | 一个深度条件变分自动编码器，它对给定错误程序的各种修复进行采样，为了解释模糊性和固有的代表性数据集的缺乏，提出了一种新的正则化器，鼓励模型生成不同的修复。 |  |  |
| 12 | Detecting '0-Day' Vulnerability: An Empirical Study of Secret Security Patch in OSS | 2019 | DSN  CCF B | 源代码 | 流行的SSL库： OpenSSL、LibreSSL和BoringSSL | 集成五种分类器的投票算法来区分安全补丁和非安全补丁。使用代码克隆识别机制来发现类似类型操作系统中的类似补丁或漏洞. |  |  |
| 13 | GGF: A Graph-based Method for Programming Language Syntax Error Correction | 2020 | ICPC  CCF B | 图和序列 | 学生C程序DeepFix和 Codeforce | 基于图形的语法修复（GGF），以帮助程序员定位和修复语法错误。GGF将代码视为令牌序列和图形的混合体。这些图基于抽象语法树（AST）结构信息。GGF使用其子AST结构对错误代码进行编码，使用指针网络预测错误位置，并生成正确的令牌。 |  |  |
| 14 | Graph-based, Self-Supervised Program Repair from Diagnostic Feedback | 2021 | ICML | 图 | （DeepFix数据集）和纠正程序合成的输出（SPoC数据集） | 首先，引入一个程序反馈图，将源代码中与程序修复相关的符号和诊断反馈连接起来，然后应用图神经网络来建模推理过程。其次，提出了一种程序修复的自监督学习，利用在线可用的未标记程序创建大量额外的程序修复示例，使用这些示例对模型进行预训练。 |  |  |
| 15 | HOPPITY : LEARNING GRAPH TRANSFORMATIONS TO DETECT AND FIX BUGS IN PROGRAMS | 2020 | ICLR | 图 | JS， (src buggy ,src fixed ) | 给定一个由图形结构建模的错误程序，我们的模型会做出一系列预测，包括错误节点的位置和相应的图形编辑，以生成修复。  针对的是更为多样和复杂的缺陷（即需要添加或删除语句才能修复的缺陷） | 扩展到由多个文件之间的相互依赖性引起的bug，或者需要多个步骤来修复的bug。 |  |
| 16 | Application of Seq2Seq Models on  Code Correction | 2021 | Frontiers in Artificial Intelligence | 源代码 | 编程语言修正  SARD | 在seq2seq模型中引入了金字塔（pyramid）编码器，显著提高了计算效率和内存效率，同时实现了与非金字塔模型类似的修复率。在ITC基准示例（只有685个代码实例）上执行了错误类型分类任务，使用迁移学习，模型在Juliet测试套件上预训练，编程语言纠正。 | 并行编码器/解码器、tree2tree模型等 |  |
| 17 | SEQUENCE R: Sequence-to-Sequence Learning for End-to-End Program Repair | 2021 | IEEE TRANSACTIONS ON SOFTWARE ENGINEERING | 源代码 | CodRep，Java数据集和Bugs2Fix数据集 | 基于令牌序列上的序列对序列学习来修复错误的方法。使用复制机制来克服源代码中词汇不受限制的问题。 | 对于修改连续代码行（即hunk），学习生成多行代码作为输出，并使用hunk的特殊标记（即<START\_BUG>和<END\_BUG>）；（ii）对于修改不同位置的多行的修复，生成一组有限的程序组合，其中包含每个可疑位置的预测固定行。  关于树到树转换学习适合于代码视图的稀疏树。 |  |