第50卷第8期 中国科学技术大学学报 Vol. 50, No. 8
2020年8月 JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA Aug. 2020
文章编号:0253-2778(2020)08-1048-10

高教程序代码作业抄袭检测的方法研究与实践

于 俊[1,2],李雅洁[2],程礼磊[2],连 顺[3],谭 昶[2],丁德成[3],刘 淇[1]
(1.中国科学技术大学计算机科学与技术学院,安徽合肥 230027;2.科大讯飞股份有限公司,安徽合肥 230088;
3.南京谦萃智能科技服务有限公司,江苏南京 210019)

摘要:学生的编程水平直接反映技术类课程的学习效果,因此教学考察中程序代码作业的比重也越来越大.由于程序代码作业抄袭成本低,导致抄袭现象不同程度地存在于各高校教学中,严重影响了学生能力的培养和教师教学的效果,打击学生学习的积极性乃至损坏学风.为此以智能且自动化方式找出学生作业的相似之处,分析学生抄袭的总体情况为目的,将人工智能算法和数据处理分析技术相结合,提出一种学生作业抄袭检测方法.首先,分析学生提交的程序代码作业的复杂情况,设计作业数据预处理流程.然后,具体提出了基于KR和Winnowing的程序代码作业相似度检测算法,与传统检测方法相比通过代码格式化等改进手段提升了学生作业相似检测的精准度,并在大批量作业检测实践中,研究优化算法增加了不同学生之间作业相似结果的区分度.为了验证相似度计算部分的有效性和实用性,进一步设计了相关的模拟实验流程(包括与JPlag检测系统的对比),给出在相同实验数据集上不同抄袭类型下的相似度计算结果.最后,依托于科大讯飞博思智慧在线学习平台对该研究进行了真实场景的实际应用.实验结果以及实际应用都表明,该程序代码作业抄袭检测方法,对高校学生程序代码作业相似度检测有效,具有很高的应用价值.

关键词:程序代码抄袭检测;相似度检测;在线智慧教育
中图分类号:TP391 **文献标识码**:A doi:10.3969/j.issn.0253-2778.2020.08.002

引用格式:于俊,李雅洁,程礼磊,等.高教程序代码作业抄袭检测的方法研究与实践[J].中国科学技术大学学报,2020,50(8):1048-1057.
YU Jun, LI Yajie, CHENG Lilei, et al. Research and practice of plagiarism detection in program code assignments by college students[J]. Journal of University of Science and Technology of China, 2020, 50(8):1048-1057.

Research and practice of plagiarism detection in program code assignments by college students

YU Jun[1,2], LI Yajie[2], CHENG Lilei[2], LIAN Shun[3], TAN Chang[2], DING Decheng[3], Liu Qi[1]
(1. *School of Computer Science and Technology, University of Science and Technology of China, Hefei* 230027, *China*;
2. *USTC iFLYTEK Co., Ltd., Hefei* 230088, *China*; 3. *Nanjing Qiancui Intelligent Technology Service Co., Ltd, Nanjing* 210019, *China*)

Abstract: The programming ability of students directly reflects the learning effect of technical courses. The proportion of program code assignments are increasing in teaching evaluation. The low cost of plagiarism of program code homework leads to the widespread plagiarism in colleges and universities, which seriously affects the cultivation of students' ability and the effect of teaching. To this end, a method for homework plagiarism detection is proposed by combining the artificial intelligence algorithm with data processing analysis technology to detect similarities in students' homework intelligently and automatically, and analyze the overall situation of plagiarism. First, the complex situation of the program code assignments submitted by students is analyzed, and the data pre-processing process is designed. Then, the similarity detection algorithm for program code assignments based on KR and Winnowing is specifically proposed. Compared with the traditional detection methods, the accuracy of similarity detection in students' homework is improved by such means as code formatting. In the practice of large-scale homework detection, the research optimization algorithm increases the differentiation of similarity results in different students' homework. To verify the validity and practicability of the core similarity calculation part of this paper, a relevant simulation experiment process (including the comparison with JPlag detection system), was designed and the similarity calculation results were given under different plagiarism types on the same experimental data set. Finally, based on iFLYTEK's Bosi intelligent online

收稿日期:2020-06-05;**修回日期**:2020-06-24
基金项目:国家自然科学基金(61922073),中央高校基本科研业务费专项(WK2150110021)资助.
作者简介:于俊,男,1981年生,博士生/工程师.研究方向:数据挖掘、用户画像、知识图谱.E-mail:ustcyujun@163.com
通讯作者:刘淇,博士/教授.E-mail:qiliuql@ustc.edu.cn

learning platform, the research has been applied in real scenarios. The experimental results and practical application results show that the proposed detection method has high validity and application value in the detection of similarities in program code assignments by college students.

Key words: plagiarism detection for program code; similarity detection; online wisdom education

0　引言

随着大数据、人工智能等技术的发展，高等院校教学也深受影响[1-2]，随之增加编程语言、数据结构与算法等信息技术类课程，教师在授课过程中更加注重理论和实践的结合[3]，程序代码作业的比重越来越大，而学生的编程水平直接反映学生对信息技术类课程的学习效果，高校也越来越多地结合大数据挖掘、统计学等相关技术及时对在线教师和学生作出一定的教育评估反馈[4]. 由于程序代码作业比重加大，并大多数以电子文档的方式提交或者是以在线做题的方式完成，学生通过开发工具和电子文档编辑器可以很容易地复制和修改电子文档，使得抄袭代码外观与原来的代码不相同，而程序结构以及运行结果却与源代码完全一样. 由于电子文档带来的抄袭技术门槛低，所以程序代码作业抄袭现象更为严重，且教师通过人工很难直接发现抄袭行为[5]. 程序代码抄袭指利用抄袭手段将一段程序代码变换为另一段程序代码，并使用抄袭检测技术检测出它们之间相似度大于某个阈值. 其中常见抄袭手段包括：直接拷贝文件，逐字拷贝，更改注释，更改变量名、方法名、类名，改变代码块顺序，增加或删除冗余语句和变量以及改变表达式中操作符和操作数顺序等. 抄袭严重影响了学生能力的培养和教师教学的效果. 首先，教师无法对学生的真实水平做出一个准确评估和评价，且无法从作业和考试中获得真实的教学反馈信息；其次，会严重打击那些认真独立完成作业的同学的积极性；最后，会损害学风，与高等院校培养学生能力的目标背道而驰.

本文根据高等院校教学的实际需求，提出了一种能够及时、自动找出学生程序代码作业的相似之处的高教程序代码作业的抄袭检测方法. 检测对象是同一个班级同一次作业学生在学习平台提交的程序代码作业. 一方面，在理论创新性上，本文对传统的抄袭检测算法进行了优化；在后续增加作业区分度的实践中，通过在相似度计算流程中加入卷积神经网络(CNN)的优化算法[6-8]，提高了不同学生之间作业相似结果的区分度，提升了学生程序代码作业相似检测的精准度. 另一方面，本文的方法实现了在学生程序代码作业复杂场景检测算法的落地，为教师判断学生程序代码作业是否抄袭提供了判断依据，同时可供教师了解学生程序代码作业是否存在抄袭、抄袭的程度，制定针对性的政策，提高学生实践能力、提升教学效果，并形成良好的教学氛围.

1　学生程序代码作业抄袭检测方法分析

1.1　抄袭检测研究现状

有关程序抄袭检测的研究工作在国外开展较早. 例如，Baxter[9]提出了使用抽象语法树来查找克隆代码，并且对比了抽象语法树的特点和单词序列的特点，指出使用抽象语法树来进行匹配的优点；Baxter 同时给出了利用抽象语法树查找克隆代码的基本算法步骤以及优化子树匹配效率的算法. Koschke[10]提出一种抽象语法树的改进算法：①把源代码生成抽象语法树；②对抽象语法树的每一个结点就进行遍历，进行序列化，并根据每一个结点生成的序列构建后缀树；③两两比较由源代码生成的后缀树，计算源代码相似度或者查找克隆代码. 该算法主要改进了子树匹配的效率.

值得重点提及的是 Winnowing 算法[11]，这是由 Schleimer 等受 Rabin-Karp 算法启发，提出的基于数字指纹的匹配算法. Rabin-Karp 算法是由 Rabin 和 Karp[12]提出的字符串匹配算法，它的基本思想是基于映射理论和素数理论定义一个称为“指印”[13]的函数，它首先将模式串映射成一个比模式串短得多的指印(位串数据)，然后将正文串中每一个长度为 m 的子串也映射成一个比子串本身短得多的指印. 要求所构造的指印函数尽可能扫描整个文本串，并且能迅速计算出每个长度为 m 的子串的指印，以精确识别文档复制问题，并应用于在线服务网站 MOSS 上的抄袭识别.

国内的相关研究包括：张文典[14]使用属性度量技术开发了国内第一个抄袭检测系统，朱江[15]将 XML 引入本领域，王继远[16]采用基于 XML 的结构度量方法进行相似度计算，赵长海等[17]提出了编译优化和反汇编的程序相似性检测方法，张鹏[18]用本体粗糙集的方法对程序代码进行了相似度度量，熊浩[19]提出了一种基于静态词法树的程序相似性检测方法；王春辉[20]对最长公共子串(GST)算法和其改进算法 RKR-GST 算法进行了研究，通过实验表明该算法的效果比使用 LCS 算法和编辑距离来计算相似度的效果要好.

现有的检测算法及抄袭检测系统虽取得了一定的成果，但仍存在着诸多不足，如多数系统只提供一种检测算法，由于不同的检测算法适用范围不同，用户不能根据自身需要选择合适的检测算法；实际检测中，大量重复的变量声明语句及抄袭者对源代码的改动等会给检测带来噪音，影响相似度计算，因此从技术角度出发，抄袭检测仍有大量可以改进之处值得研究.

1.2　问题解析

本文针对高等院校教学中存在不同程度的抄袭现象，做了实际的调研分析. 目前高校对于该问题的主要应对措施，一是通过教师主观判断是否存在抄袭现象，该方法存在一些缺点：①教师批改作业兼顾发现是否抄袭，注意力分散，易于造成漏判；②教师人工鉴别抄袭行为效率低下；③学生通过修

改变量名、改变结构，增加或删除冗余语句和变量等操作难以被发现；二是通过代码相似度来评价学生程序代码作业是否抄袭，取值区间为[0，1]，取值为 1 时表示两份代码完全相同，取值为 0 时表示两份代码完全不同. 通常计算作业相似度的方法主要包括以下几种：(Ⅰ)根据 Halstead 科学软件度量理论，采取基于属性计数的检测技术评价代码相似度. 由于基于属性计数的方法过于抽象，忽略了程序代码的结构信息问题，修改了源代码中的条件语句等结构，采用这种方法检测抄袭易失败. (Ⅱ)使用编译优化和反汇编技术将源代码转变为目标代码，然后通过删除和替换汇编指令中对程序特征影响不大的元素，使用决策函数计算程序代码的相似度，但是这种方法要求代码必须是可编译的，并且不适合大规模应用. (Ⅲ)使用基于 XML 方法的结构度量方法进行相似度的计算. 这种方法使用的特征维度较高，导致计算速度较慢. (Ⅳ)使用 LCS 算法和编辑距离来计算作业相似度. 这种方法对更改代码块顺序的检测效果不够好.

本文在现有抄袭检测手段的基础上，根据存在的问题，调研大量的学生程序代码作业数据，对相似度计算方法做了改进：为避免学生通过修改变量名、改变结构、增加或删除冗余语句和变量等操作逃避检测，对学生程序代码作业数据进行代码格式化操作；为处理变量重命名，代码重排等问题，选取 KR 算法计算 Hash 值，使用 Winnowing 算法计算作业相似度，实现学生程序代码作业相似度的高效检测. 同时，针对不同学生作业之间相似结果区分度不大的情况，使用 CNN 进行代码文本特征抽取，利用前馈网络模型替代复杂的标记串处理与字符串匹配，进行相似度计算，改进学生程序代码作业的区分度，提升检测的精准度. 实验结果表明，本文提出的学生程序代码作业抄袭检测方法，对高校学生程序代码作业相似度检测高效，具有很高的应用前景.

2 主要工作

2.1 检测流程设计

根据高等院校学生作业抄袭检测的实际场景，设计本文作业抄袭检测方法的流程，如图 1 所示.

首先，考虑到学生提交作业的文档情况复杂，存在着大量的读取错误、不符合作业格式规范要求以及重复提交等无效文件，因此需要有作业文件的过滤提取环节，过滤掉不符合检测文件要求的作业. 针对存在直接复制文件的抄袭手法，在对作业文件进行正式的相似度计算前设计简单抄袭判断行为模块可以节省开支，理论上可以认定文件属性完全一致的两份作业属于完全抄袭. 然后，提取上一环节结束生成的待检测数据. 待检测数据在相似度计算开始前，需要进行数据预处理，输出可供检测数据，开始进入相似度计算环节. 最后，在得到初步相似度计算结果之后，对结果进行分析，发现教师布置作业简单或者程序共同依赖某些功能定义和使用基础赋值语句的情况，会导致班级学生作业

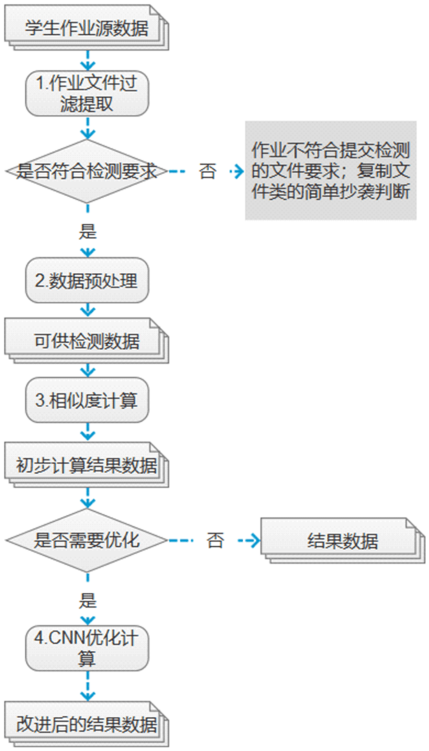


图 1 学生程序代码作业相似度计算流程

Fig. 1 Student assignments similarity calculation process

的相似度普遍较高. 针对不同学生之间相似结果区分度不大的情况，使用 CNN 算法改进，优化相似度计算结果.

2.2 学生程序代码作业抄袭检测框架

(Ⅰ)作业文件过滤提取与预处理

由于学生程序代码作业过滤工作量大，需要反复实验，本文设计出一套实用的过滤规则，以支撑整个系统. 学生打包上传的作业不尽相同，因此获得学生上传的课程作业后，首先要对作业压缩包进行解压，找到最终作业文件. 然后对压缩包里的文件进行过滤，如对不属于代码的作业文件、重复提交的作业进行过滤等. 允许上传的类型就是设置待检测的类型. 最后得到过滤后文件，判断是否是简单抄袭，如果不是，作业文件由此进行预处理. 预处理需要读取作业文件内容，因为算法对代码长度有要求，对空文件或者内容过少的代码作业进行提示. 另外，针对学生某次作业提交多份供检测代码作业的情况，可将多份作业进行拼接，形成一个完整的作业数据.

(Ⅱ)相似度计算

基于高校学生程序代码作业的计算步骤如图 2 所示. 首先对作业代码进行格式化，接着提取格式化后代码的 Hash 特征，然后使用 Winnowing 算法进行 Hash 值的筛选建立文件指纹集，最后计算两份代码文件的指纹重复度得到其相似度.

(Ⅲ)相似学生分组

本文在不同学生作业相似度计算的基础上，为了丰富学生程序代码作业抄袭检测的具体内容，而不只是给出简单的重复率，增加了对于疑似抄袭团

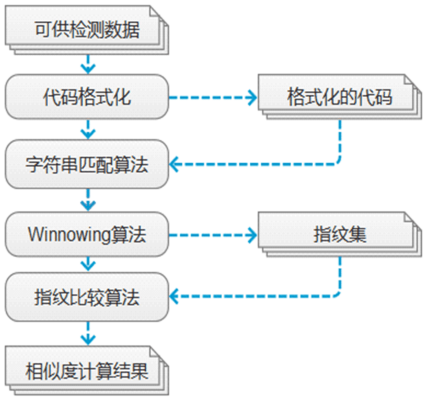


图 2　学生程序代码作业相似度计算步骤

Fig. 2　Student assignments similarity calculation steps

伙的跟踪. 在对任意两个学生程序代码作业计算相似度后，一个班级的所有学生的相似情况就可以生成一个矩阵，根据重复度挑选相关的学生分组，可以给出具体的疑似抄袭团伙.

(Ⅳ)相似内容位置定位

本文设计了相似内容位置定位环节，能够更直观地反映学生程序代码作业相似检测的具体内容. 使用标记串 token 记录源代码位置和代码格式化后的位置，能够保留作业原位置的信息和代码格式化后的位置信息. 通过指纹匹配对应的原位置和格式化后的位置，定位学生程序代码作业的相似位置.

(Ⅴ)学生作业相似计算结果区分度优化

在作业相似计算后续的实践中，我们发现，在大批量学生作业的检测中，往往会存在学生作业相似结果区分不大的情况，这时候抄袭检测效果会受影响. 为了优化相似度计算结果，本文使用 CNN 进行代码文本特征抽取，利用前馈网络模型替代传统方法中复杂的标记串处理与字符串匹配，进行相似度计算，改进学生程序代码作业相似度计算结果，如图 3 所示.

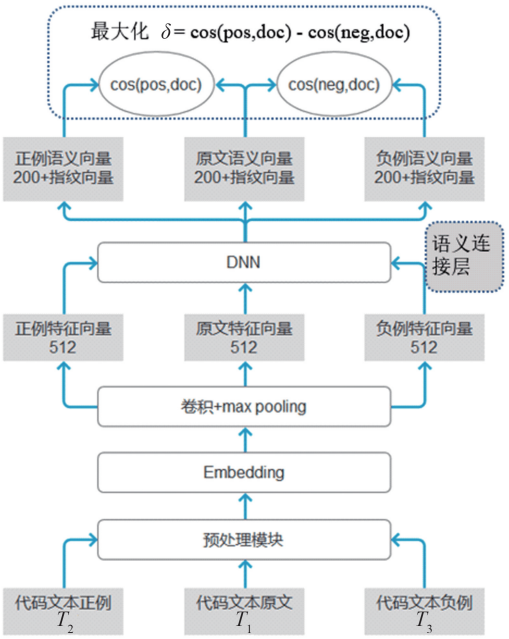


图 3　基于 CNN 模型改进的学生作业相似度计算流程

Fig. 3　Improved calculation process of student assignments similarity based on CNN model

3　核心技术研究

3.1　代码格式化

代码格式化主要分为：属性度量和结构度量.

属性度量就是将代码文本表征为向量、代码行数、常量和变量个数，不同操作符、不同操作数的个数等都是向量的维度. 由于属性度量中的特征抽取仅考虑代码所包含的属性特征，而忽略了代码的组织结构信息，这使得属性计数法在“少量抄袭，创新性重组织代码”的检测实证中往往表现的不尽如人意. 为此，Verco 与 Wise[21] 提出了增加属性统计维度的观点，但也无法从根本上改善属性计数法效果不佳的现状.

结构度量则是将程序代码格式化为特定标记字符串，主要有基于文本、标记串、语法树等，然后再用字符串匹配的方法计算相似度. 结合本文方案，我们将重点分析结构度量. 结构度量法的流程分为两步：第一步，特征抽取阶段使用一串标识串来表示程序代码. 第二步，以度量标识串的相似度代替空间向量的度量. 在结构度量算法中，重点便是代码转换成标识串的效果与标识串相似度度量的算法，因为在该算法中，能否将代码在控制流、嵌套循环结构的相似性反映在具体数据上是重要的评判标准.

第一阶段的代码转换的方法众多，基本可以分为基于 tree 的方法[22]、基于 graph 的方法、基于 token 的方法[23]、基于 metrics 的方法[24]、基于字符串的方法[25]、混合型方法等几大类. 第二阶段的标记串度量阶段，常用的方法有 5 种：GST[26]、序列匹配、最长公共子串、RKR-GST[27]、点阵图[28]. 除此之外，还有少量的基于图像[29]对比的检测方法.

通过对代码格式化方法进行表 1 所示的比较. 我们选取基于标记串的方法，将空格和注释过滤掉，所有的变量名替换为 V，所有字符串替换为 S，所有函数名替换为 F.

3.2　字符串匹配算法

字符串匹配是判断一个串是否是另一个串的子串，对于待匹配的两个字符串，把两者中较短的字符串作为模式串，用 P 表示，有 $P=\{p_1,p_2,p_3,\cdots,p_m\}$；将较长的字符串作为文本串，用 T 表示，$T=\{t_1,t_2,t_3,\cdots,t_m\}$. KR 算法是一种随机串匹配算法，该算法把长度为 m 的模式串按照某个哈希函数计算得到一个对应的散列值，这个函数称为散列函数，把文本串依次切割为 $n-m+1$ 个长度为 m 的子串，也按照这个散列函数计算得到相应的散列值. 将模式串的散列值与文本串的所有散列值逐一比较，找出所有与模式串具有相同散列值的文本子串，再判断其对应的字符是否相等，决定该模式串是否是文本串的子串.

选择基于标记串的代码格式化方法之后，我们选择随机串匹配(KR)算法计算 Hash 值，具体内容如表 2 所示[30].

表 1 代码格式化方法比较

Tab. 1 Code formatting method comparison

分类	描述	检测准确性	抗混淆能力	时空复杂度
基于属性统计	操作符与操作数的量	低:过度抽象	基本不抗混淆	统计特性提取开销小
基于文本	每个代码行视为字符串	中:依赖于公共子串计算	基本不抗混淆,对改变文本顺序、增加冗余敏感	计算开销小
基于树	语法树	中:依赖于树的精炼度	考虑语法及结构,但难以对抗代码重排、语句拆分	树的构建成本高
基于图	语法图	高:依赖于图的精炼度	考虑语法和语义特征	PDG 构建代价很高,字图匹配 NP 问题
基于标记串(token)	替换成一系列标准表达	中:依赖于 token	能抵抗变量重命名、代码重排等混淆,但难应对冗余代码植入	文本结构及词法分析为主,解析速度快

表 2 字符串匹配算法

Tab. 2 String matching algorithm

分类	描述
最长公共子序列(LCS)	长度最长的相同字符序列,严格有序
贪心字符串匹配(GST)	最大匹配与最小匹配长度,无序
随机串匹配(KR)	"滚动"哈希检索
KRGST	长度为 m 的模式串散列值与 $t-m+1$ 个子串散列值匹配

3.3 学生程序代码作业指纹生成算法

首先,由第 2 节流程设计可知,针对存在直接复制文件的抄袭手法,在对作业文件进行正式相似度计算前,可以设计直接复制文件这种简单的抄袭行为判断算法,因此本文引用计算 Sha-1 值的方法.

Sha-1 是一种密码散列函数,它可以生成一个被称为消息摘要的 160 位(20 字节)散列值,散列值通常的呈现形式为 40 个十六进制数.一份文件的 Sha-1 值通常也称为数字签名,对文件内容进行计算后可以得到 Sha-1 值;如果两个文件的 Sha-1 值相同,那就能断定是完全相同的文件,因此也可以直接判定抄袭.这样可以高效地筛选出简单抄袭的作业文件,节省计算资源.

对于其他作业文件的相似度计算,本文根据 KR 算法生成的 Hash 值,选取 Winnowing 算法基于数字指纹来完成代码的特征提取,以精确识别代码重复问题.

下面举一个例子,解释 winnowing 算法过程:

```
X ← 'def finger():  print 1' # 原句字符串
for eachin X:
if each = 空白、注释:
delete each# 标记化
Y ← 5_gram_def(X)  # 对 X 取 5-gram 集合
hash value ← hash(Y)  # 对 Y 取 Hash 值集合
w_value= Winnowing(hash value, w=4) # 在 Z 上滑动窗口,每个窗口取 4 个值
for each in w_value:
x=min(each) # 每个窗口选择最小值
position(each) add in location_set #记录位置
x add in Document_fingerprint # 生成的文本指纹
return Document_fingerprint
```

以此得到文档的指纹,可以进行相似度计算以及关于相似位置内容定位.

3.4 结果计算与相似位置内容定位

假设有两个同学 T_1,T_2,其代码作业文件通过前面的算法提取出了文档指纹集合,分别记为 X 和 Y.通过计算两个代码文件的相同的指纹个数,分别除以二者文档指纹集合的指纹个数,就得到两个文档的相似度情况.这里有两种计算结果:

T_1 相对于 T_2 的相似度为

$$T_1T_2=\frac{X\cap Y}{X}.$$

T_2 相对于 T_1 的相似度为

$$T_2T_1=\frac{X\cap Y}{Y}.$$

因为 T_1,T_2 本身的学生程序代码作业文件内容量不同,故计算公式的分母不同,导致二者的相似度有两个值.

对于相似位置内容的定位问题,如对于 token:('def', 74, 1)(其中第 2 个参数是源代码位置,第 2 个参数是在格式化代码中的位置),本文采用如下方法:

使用 k-Grams 记录哈希值和在格式化代码中位置的起点和终点,如 k-grams:('defF',30434,0,5),k-grams,哈希值,格式化代码中的起点、终点.

如果指纹匹配,先将 k-grams 在格式化代码中的位置信息和 token、格式化代码中的位置信息对应,然后获取 token 中的原始文件的起始位置和终点位置.

3.5 CNN 模型优化计算结果

使用 Winnowing 算法获取样本区分度不大时,为了增加区分度需要进一步使用 CNN 模型优化相似度计算结果,步骤如下:

Step1 使用 Winnowing 算法获得种子样本,设置相似度大于 80%时记为正样本 1,相似度小于 50%记为负样本 0;选取完全不相似的 1000 个样本为种子数据,用常用的抄袭手段对样本 1∶15 进行抄袭样本构建,产生 15000 对正样本;种子样本两两匹配,选取 30000 对为负样本;

Step2 使用词法分析器 Pygment 预处理代码;

Step3 将原文 T_1、正例 T_2、负例 T_3 的 3 元输入，把代码变成矩阵，利用矩阵还原学生程序代码作业，使用 TF-IDF 值进行填充；

Step4 使用 CNN 模型的 512 个 filter 模拟 k-gram 的卷积过程，然后进行池化，得到 F_1，F_2，F_3 特征向量；

Step5 叠加一层 DNN 模型，得到全部 200 维度的语义向量，并在此基础上，拼接 Winnowing 算法得到的指纹向量.

Step6 使用目标函数：$\max(0,\mathrm{simi}(F_1,F_2)-\mathrm{simi}(F_1,F_3)+1)$ 重新计算作业相似度，提升学生作业相似检测的精准度.

4　相似度计算实验分析

为了验证本文相似度计算的有效性和实用性，我们设计了相关的模拟实验流程，并给出了实验结果，包括实验对象设计，相似度度量值的可信性和弹性评估以及对比分析结果等. 其中，我们在相同实验数据集上对不同抄袭类型的相似度计算结果，与 JPlag 检测系统（一款大众熟知并在实践中被成功地用于检测学生程序提交中的剽窃行为的系统）进行对比并给出实验结果，以说明本文的系统和方法的有效性和实用性.

4.1　实验对象设计

LeetCode 网站具有海量编程题库，适用于真实的日常技术开发和学习场景，按照题目考察点分类，和教师布置的作业类似. 为了模拟作业的真实性，实验选取 LeetCode 网站上的数据结构编程题，分别选取题目分别是 4 号题“寻找两个有序数组”、5 号题“最长回文子串”、980 号题“不同路径”. 将 3 道程序编程题的 python 题解包装为 3 次 python 程序代码作业的实验数据集. 其中，每次作业包含 10 个不同版本的程序作业，总共 30 份原始作业，我们称为数据集 A. 通过人工校对，保证实验数据集 A 的作业是不存在抄袭的情况. 接下来我们将设计生成抄袭作业数据集.

基于上述 30 份作业，总结分析抄袭手段并采用包括有完整拷贝、修改代码块顺序、修改注释、添加冗余代码和更改控制结构在内的 5 种抄袭手段，修改原始作业的程序代码进而生成抄袭对，得到了 150 份抄袭作业样本，并按照抄袭类别分类，我们称为数据集 B.

4.2　相似度度量值的可信性和弹性评估

这里引用 Collberg 对可信性（credibility）和弹性（resilience）两个基本特性的描述[31].

（Ⅰ）可信性：对于不同版本的代码 P 和 Q，利用系统计算两者相似度 Similarity(P,Q)，如果 Similarity(P,Q)$<\varepsilon$，则认为该系统具备可信性.

（Ⅱ）弹性：假定代码 Q 是通过模拟抄袭手段对程序 P 生成的抄袭版本，利用系统计算两者的相似度 Similarity(P,Q)$\geqslant 1-\varepsilon$，则任务系统对于抄袭手段具备弹性. 综上可知，可信性要求版本不同的作业总不相似，弹性要求系统识别抄袭的能力.

对于数据集 A，有 30 份不同版本的作业，任一两份作业形成一对程序，共需要检查 435 对程序，用本文检测方法计算相似度，统计数据集 A 中两两匹配的作业对相似分布情况如图 4 所示. 由图 4 可知，不同版本作业的相似性普遍很低，都在 40% 以下. 其中有少数几个程序对相似性在 40% 以上，甚至高于 60%. 通过人工检查发现，作业题目简单，代码量较少时，程序共同依赖某些功能定义和基础赋值语句类似，以及作业对空间和时间的要求导致某些算法不会被任何解法使用，导致系统在处理这些程序对得出较高相似度度量值的原因.

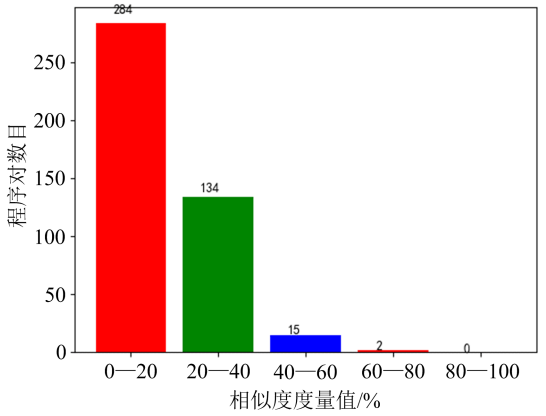


图 4　数据集 A 两两匹配的相似度分布

Fig. 4　Similarity distribution of pairs of data set A

将数据集 B 与原始代码形成抄袭对，共需检查 150 对程序. 用本文检测方法计算相似度，统计抄袭对相似分布情况如图 5 所示. 所有程序对相似度值都大于 60%，93% 的程序对的相似度值大于 80%，这表明系统对抄袭手段具备很好的检测效果.

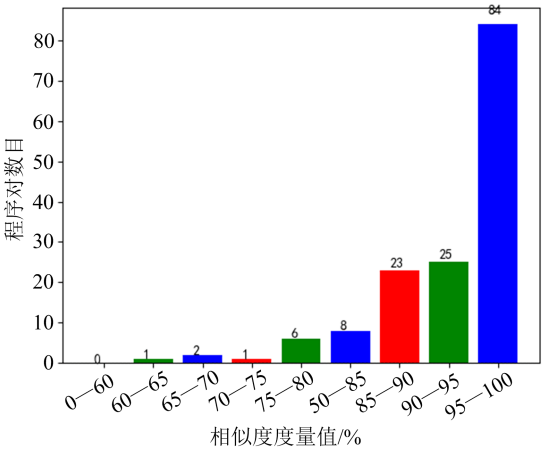


图 5　抄袭对相似度分布

Fig. 5　Similarity distribution of plagiarism pairs

4.3　对比分析

（Ⅰ）与其他查重系统的对比分析

数据集 B 是根据数据集 A 修改得到的，与原始程序形成 150 对抄袭对. 表 3 给出了本文介绍的方法与 JPlag 对抄袭对的相似度度量的平均情况.

由表 3 可知，随着抄袭手段复杂度的提高，相似度度量值有所下降，特别是在更改控制结构时，相似度下降尤为明显. 本文检测方法和 JPlag 在完整

拷贝、修改注释结果之间差距不大;在修改代码块顺序、添加冗余代码和更改控制结构抄袭手段上,本文检测方法重复度会比 JPlag 高,更能检测出抄袭手段. 这是因为本文检测方法在进行程序代码格式化为特定标记字符串后,对长度不小于 k 的子串进行匹配并对指纹集去重,达到噪声抑制和位置无关的效果.

表 3 与原始程序相似度度量值

Tab. 3 Similar degree to the original program

程序抄袭种类	与原始程序相似度度量值	
	本文检测方法	JPlag
完整拷贝	100.0%	100.0%
修改代码块顺序	92.1%	71.2%
修改注释	99.6%	100.0%
添加冗余代码	94.6%	74.1%
更改控制结构	83.2%	60.5%

(Ⅱ)检测效果的对比分析

学生程序代码作业抄袭检测效果的评估采用 3 种指标,包括 URC(union of resilience and credibility)[32],F_1-score, MCC(Matthews correlation coefficient)[33].

为了方便表达评价指标的计算公式,先对一些数据集进行定义.

EP,由确定抄袭关系的代码对组成的集合,即当 $(P,Q)\in \mathrm{EP}$,则 (P,Q) 是确定存在抄袭关系的代码对.

JP,由被作业评估系统判定为抄袭的代码对组成的集合,即当 $(P,Q)\in \mathrm{EP}$,则 (P,Q) 是被当前系统判定存在抄袭关系的代码对.

EI,由被确定不存在抄袭关系的代码对组成的集合,即当 $(P,Q)\in \mathrm{EP}$,则 (P,Q) 是确定相互独立的代码对.

JI,由被作业评估系统判定不存在抄袭关系的代码对组成的集合,即当 $(P,Q)\in \mathrm{EP}$,则 (P,Q) 是被当前系统判定相互独立的代码对.

基于以上定义,URC 可由下式计算

$$\mathrm{URC}=2\times\frac{R\times C}{R+C}.$$

式中,$R=\dfrac{|\mathrm{EP}\cap\mathrm{JP}|}{|\mathrm{EP}|}$ 表示所有确定抄袭关系的代码对中被系统正确判定的比例,$C=\dfrac{|\mathrm{EI}\cap\mathrm{JI}|}{|\mathrm{EI}|}$ 表示所有确定不存在抄袭关系的代码对中被系统正确判定的比例. URC 是这两个比例数值的调和平均数.

在 F-measure 评估系统中,我们选取最常用的评估值,准确率(Precision)和召回率(Recall)的调和平均数 F_1.

$$F_1=\frac{2\times\mathrm{Precision}\times\mathrm{Recall}}{\mathrm{Precision}+\mathrm{Recall}}.$$

式中,$\mathrm{Precision}=\dfrac{|\mathrm{EP}\cap\mathrm{JP}|}{|\mathrm{JP}|}$ 表示所有被系统判定抄袭关系的代码对中被确认存在抄袭关系的代码对的比例,即评价系统的准确率,$\mathrm{Recall}=\dfrac{|\mathrm{EP}\cap\mathrm{JP}|}{|\mathrm{EP}|}$,即评价系统的召回率,故

$$\mathrm{MCC}=\frac{\mathrm{TP}\times\mathrm{TN}-\mathrm{FP}\times\mathrm{FN}}{\sqrt{(\mathrm{TP}+\mathrm{FP})(\mathrm{TP}+\mathrm{FN})(\mathrm{TN}+\mathrm{FP})(\mathrm{TN}+\mathrm{FN})}}.$$

式中,$\mathrm{TP}=|\mathrm{EP}\cap\mathrm{JP}|$(真正)表示确定抄袭关系的代码对中被系统判定为抄袭关系的代码对的数量,$\mathrm{FN}=|\mathrm{EP}\cap\mathrm{JI}|$(假负)表示确定抄袭关系的代码对中被系统判定为不存在抄袭关系的代码对的数量,$\mathrm{FP}=|\mathrm{EI}\cap\mathrm{JP}|$(假正)表示被确定不存在抄袭关系的代码对中被系统判定为抄袭关系的代码对的数量,$\mathrm{TN}=|\mathrm{EI}\cap\mathrm{JI}|$(真负)表示被确定不存在抄袭关系的代码对中被系统判定为存在抄袭关系的代码对的数量.

对于每种度量指标而言,在同一阈值下其值越高,表明方法的检测性能越好. 图 6(a)~(c)分别给出了不同阈值下对于 3 种度量标准的对比结果. 由图 6 可以看到,在任何阈值下对于完整拷贝、修改注释和添加冗余代码的检测性能总优于更改代码块顺序和修改控制结构.

通过实验发现,当系统设置阈值为 85 时,可以很好地检测抄袭手段.

5 真实场景测试与应用

本文依托于科大讯飞在线智慧教育博思学习平台的真实场景. 通过使用基于本文方法的学生程序代码作业抄袭检测算法,以班级为单位计算学生作业的相似度检测基本结果.

5.1 基于真实场景数据的测试结果

我们从博思学习平台上获取多个高校共 828 个班级的 102 G 学生程序代码作业. 按照本文算法,计算班级内每两份作业的相似度,设置判断为抄袭的相似度阈值,计算班级抄袭人数与总人数比例,称为班级相似率,然后将 828 个班级相似率平均,得到结果如图 7,8 所示.

图 7 表示在不同阈值下班级平均相似率,随着阈值的提高,班级的抄袭情况在逐步减少. 图 8(a)~(c)展示在不同阈值下班级相似率的分布情况,随着阈值的提高,抄袭人数占总人数大于 75%的班级在减少,50%~75%和小于 50%的班级在增加.

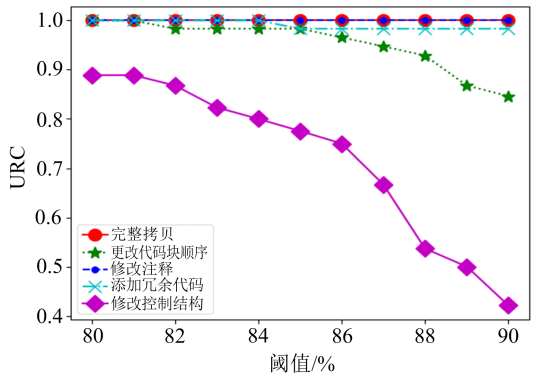
以上结果表明,程序代码类作业抄袭情况较为普遍,教师无法对学生的真实水平做出一个准确评估和评价.

5.2 算法上线效果

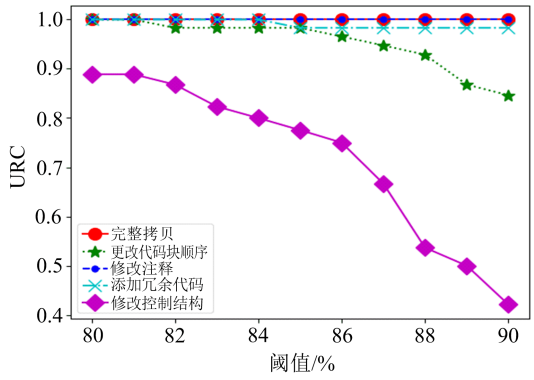
图 9 展示本文检测算法上线应用后,平台学生完全拷贝别人作业(100%抄袭)人数占总人数比例从 53.6%降为 2%,96%以上原抄袭学生放弃简单抄袭. 据反馈,学生之间通过讲解方法自主实现的方式进行作业完成. 达到以下效果:①改善了学校学习风气;②提升了教师批改作业效率,有效掌握教学情况;③减少了学生抄袭情况.

5.3 可视化精准展示班级重复情况

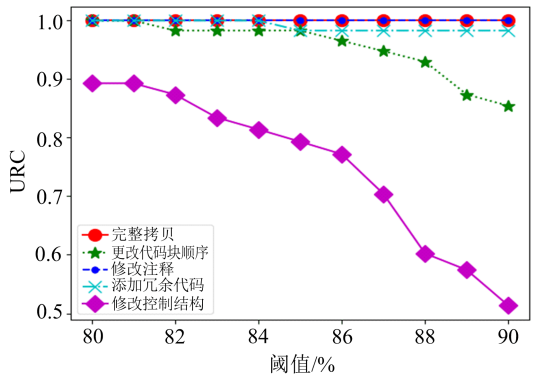
由图 10 可知,通过本文的相似度检测方法构建



(a) Comparison result based on URC



(b) Comparison result based on F1



(c) Comparison result based on MCC

图 6　3 种不同度量标准的比较

Fig. 6　Comparison of three various metrics

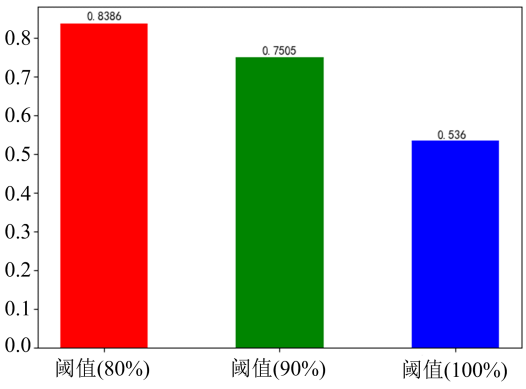
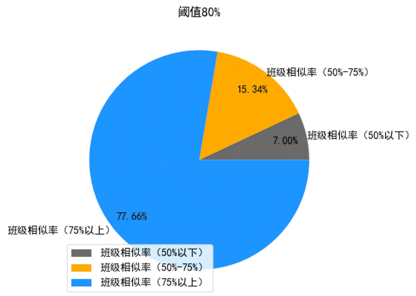


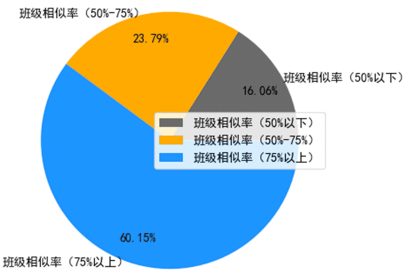
图 7　不同阈值下班级平均相似率

Fig. 7　Class average similarity with various threshold

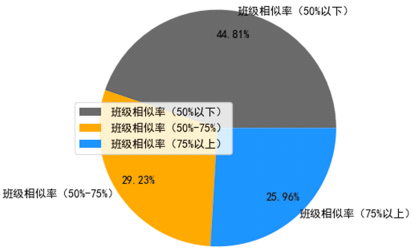
一个自动化的学生程序代码作业抄袭检测方法，能



(a)阈值80%班级相似情况分布



(b)阈值90%班级相似情况分布



(c)阈值100%班级相似情况分布

图 8　不同阈值下班级相似情况分布

Fig. 8　Class similarity distribution with various threshold

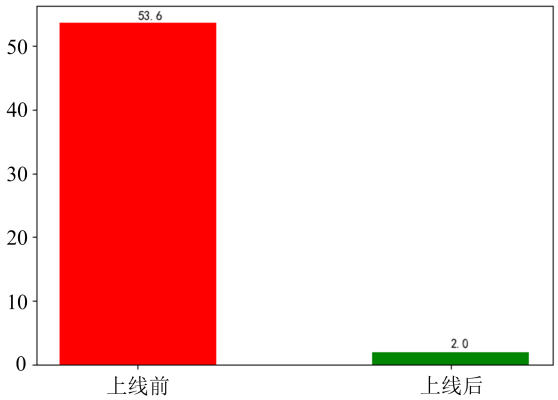


图 9　上线前后完全抄袭(100%重复)效果对比

Fig. 9　Complete plagiarism result (100% repetition) before and after applying the system

够自动去除噪音，从学生程序代码作业提取足够多的特征信息，精准地检测作业是否相似，并通过作业重复率矩阵直观地看到学生程序代码作业相似度分布情况.

图 11 展示了针对相似学生分组的检测结果，对于相似度极高的小团伙，自动标注为疑似抄袭团伙，支持一键批阅处理，能够在给出作业相似度的

基础上，进一步帮助教师分析判断是否存在疑似抄袭行为.

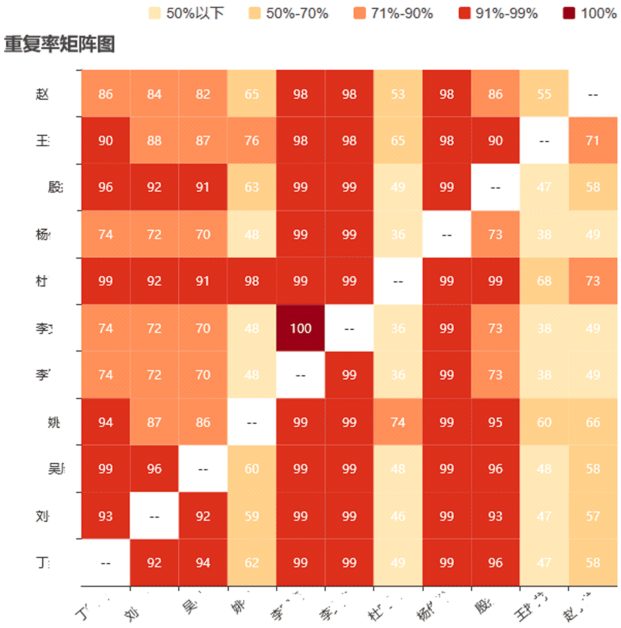


图 10 作业重复率矩阵

Fig. 10 Matrix for assignments similarity

图 11 相似学生分组

Fig. 11 Groups of similar students

实验结果显示，本文提出的学生程序代码作业抄袭检测方法，实现了自动化的作业相似度精准检测，自动去除噪音. 同时对作业相似学生分组检测的结果，便于教师分组评判、批阅和发现抄袭的深层次原因，因此本文模型对高校学生程序代码类作业抄袭检测是有效的，具有很高的应用价值.

5.4 应用优化方案

基于上述真实场景得到的作业和相似度数据，按照相似度选取完全不相似的 5000 份作业为种子数据，依据抄袭手段对样本进行 1∶15 抄袭样本构建，产生 75000 对正样本；种子样本两两匹配选取 150000 对为负样本. 按照图 3 流程得到预训练模型，作为特征提取工具.

当学生程序代码作业指纹生成算法计算的相似度都过高，无区分度时，考虑将代码输入 CNN 预训练模型中提取新特征，计算新的相似度，综合评价相似情况.

6 结论

本文针对广泛存在于高等院校教学中不同程度的抄袭现象，研究了学生程序代码作业抄袭检测方法，包括预处理时对作业自动去除噪音，改进了传统的相似度计算方法流程，实现了自动化的作业抄袭精准检测，并且在抄袭作业分组、定位学生程序代码作业的相似位置等方面也做了一定的研究. 另外，在大批量作业检测实践中，通过 CNN 模型处理增加了相似作业的区分度. 最终，本文完成了学生程序代码作业抄袭检测方法的研究和实践，从多个角度阐述了研究方法的合理性，并通过设计实现相似度计算模拟实验以及在线教育平台实际应用验证了方法的实用性和有效性.

随着高校教育教学中对学生编程实践等作业越来越重视，学生程序代码作业抄袭检测方法的优化也必将在重点研究的行列. 例如，在处理学生程序代码作业过程中最大程度上减少对预处理的依赖以及开发智能算法，自动学习代码内容和抄袭手段等方法的研究.

参考文献(References)

[1] 傅钢善. 教育技术发展轨迹探讨[J]. 电化教育研究，2005(09)：22-26.

[2] HONG C M，CHEN C M，CHANG M H，et al. Intelligent web-based tutoring system with personalized learning path guidance[J]. Computers &Education，2008，51(2)：787-814.

[3] 韦琳，袁泉，霍剑青，等. E-learning 非结构化数据管理系统的构建与实现[J]. 中国科学技术大学学报，2010，40(06)：623-628.

[4] 黄振亚，苏喻，吴润泽，等. 一种面向教育评估的智能教育辅助平台[J]. 中国科学技术大学学报，2015，45(10)：846-854.

[5] 韩冰. 基于 FTP 教学平台的代码相似度检测的研究[J]. 计算机光盛软件与应用，2012(09)：217-218.

[6] LECUN Y，BOTTOU L，BENGIO Y. Gradient - Based Learning Applied to Document Recognition[M]. Proceedings of the IEEE，1998.

[7] KRIZHEVSKY A，SUTSKEVER I，HINTON G E. ImageNet classification with deep convolutional neural networks[J]. Communications of the ACM，2012：60(2)：1-9.

[8] 殷丹平. 基于 CNN 的代码相似度检测研究与代码查重系统[D]. 北京：北京邮电大学，2018.

[9] BAXTER. Clone detection using abstract syntax trees[J]. International Conference on Software Maintenance 1998，1998：368-377.

[10] KOSCHKE R. Clone detection using abstract syntax suffix trees[J]. Working Conference on Reverse Engineering 2006，2006：253-262.

[11] SCHLEIMER S，WILKERSON D S，AIKEN A. Winnowing：Local algo-rithms for document fingerprinting[C]// Proc of the 2003 ACM SIGMOD Int'l Conf on Management of Data，2003：76-85.

[12] KARP R M，RABIN M O. Efficient randomized pattern-matching algorithms[J]. IBM Journal of Research and Development，1987：115-126.

[13] 苏德富，钟诚. 计算机算法设计与分析[M]. 北京：电

子工业出版社，2001.
[14] 张文典，任冬伟. 程序抄袭判定系统[J]. 小型微型计算机系统，1988(10)：34-39.
[15] 朱江. 基于 XML 的程序设计自动批改的研究[D]. 湘潭：湘潭大学，2005.
[16] 王继远. 一种用于软件作业评判系统的程序结构分析算法的设计与实现[D]. 北京：北京邮电大学，2007.
[17] 赵长海，晏海华，金茂忠. 基于编译优化和反汇编的程序相似性检测方法[J]. 北京航空航天大学学报，2008(06)：711-715.
[18] 张鹏，王国胤，陶春梅，等. 基于本体粗糙集的程序代码相似度度量方法[J]. 重庆邮电大学学报(自然科学版)，2008，20(06)：737-741.
[19] 熊浩，晏海华，赫建营，等. 一种基于静态词法树的程序相似性检测方法[J]. 计算机应用研究，2009，26(04)：1316-1319＋1326.
[20] 王春晖. 程序代码抄袭检测中串匹配算法的研究与实现[D]. 内蒙古呼和浩特：内蒙古师范大学，2008.
[21] VERCO K L，WISE M J. Software for detecting suspected plagiarism：Comparing structure and attribute-counting systems [C]. Australasian Conference on Computer Science Education，1996：81-88.
[22] KOSCHKE R，FALKE R，FRENZEL P，et al. Clone Detection Using Abstract Syntax Suffix Trees[C]. Working Conference on Reverse Engineering，2006：253-262.
[23] KAMIYA T，KUSUMOTO S，INOUE K，et al. CCFinder：A multilinguistic token-based code clone detection system for large scale source code[J]. IEEE Transactions on Software Engineering，2002，28(7)：654-670.
[24] ELMATARAWY A，ELRAMLY M，BAHGAT R，et al. Code clone detection using sequential pattern mining [J]. International Journal of Computer Applications，2015，127(2)：10-18.
[25] BAKER B S. Parameterized duplication in strings：Algorithms and an application to software maintenance [J]. SIAM Journal on Computing，1997，26(5)：1343-1362.
[26] Wise M J. Running karp-rabin matching and greedy string tiling [J]. Software - Practice and Experience，1993.
[27] PRECHELT L，MALPOHL G，PHILIPPSEN M. Finding plagiarisms among a set of programs with JPlag[J]. Universal Computer Science，2000，8(11)：1016-1038.
[28] GRANVILLE A. Detecting Plagiarism in Java Code [J]. Supervisor Yorick Wilks，2002.
[29] RAGKHITWETSAGUL C，KRINKE J，MARNETTE B，et al. A picture is worth a thousand words：Code clone detection based on image similarity [C]. International Workshop on Software Clones，2018：44-50.
[30] 牛永洁，张成. 多种字符串相似度算法的比较研究[J]. 计算机与数字工程，2012，40(03)：14-17.
[31] MYLES G，COLBERG C. Detecting Software Theft via Whole Program Path Birthmarks[M]. Springer Berlin Heidelberg，2004.
[32] XIE X，LIU F L，LU B，et al. A software birthmark based on weighted *k*-gram [C]//2010 IEEE International Conference on Intelligent Computing and Intelligent Systems. IEEE，2010.
[33] MATHEWS B W. Comparison of the predicted and observed secondary structure of T4 phnge lysozyme [J]. Biochim Biophys Acta，1975，405(2)：442-451.