**基于属性度量的抄袭检测**

*Edward L. Jones*

*Department of Computer and Information Sciences*

*Florida A&M University*

*Tallahassee, FL 32307*

*850-412-7362* [*ejones@cis.famu.edu*](mailto:ejones@cis.famu.edu)

**摘要**

在编程课程的抄袭是一个正在蔓延和令人沮丧的问题，大大降低了教育质量。由于难以界定团队合作和抄袭，对代码抄袭的判定是很困难的。追求检举抄袭带来的情感和法律风险，对学生和教师来说，教师承担主要责任。在本文中，我们提出了一个基于属性度量的系统，用于计算代码间相似度和收集代码抄袭的有利证据。通过发布匿名测试结果，来创造一种公开讨论代码抄袭问题的氛围。

**1 引言**

抄袭是程序设计课程中常见的问题。虽然课程教学大纲警告学生严禁抄袭，但学生总会努力地使抄袭的代码更难被检测。重要的是，教育工作者解决剽窃问题，但更多的时间和精神负担，需要面对罪犯[1]。教育工作者有责任不仅要谈论道德和社会责任问题[2]，但也查看道德行为与技术能力相同的权重。当然，抄袭计算机科学专业的学生将是一个永恒的诱惑。更适合什么样的道德困境除了抄袭，以满足课程的任务来对待道德问题吗？抄袭是整个学生的学术生涯中一个具体的道德问题将再次出现。

本文阐述了如何简单的指标可以提供初步筛查水平过度的协作和剽窃。指标的应用是一个整体的过程中积累了必要的证据，面对违规者的一部分。我们的做法是可扩展的，允许使用额外的措施和文件。该系统提供了一个消毒的报告，从监测的结果，这是发给学生，使他们了解事实，即“大哥哥”看。

在第2部分中，我们讨论抄袭的预防，检测和对抗的其他方法。在第3节中，我们描述了我们的方法所使用的指标，对可疑程序的环境中收集证据。第4节中包含的样本结果，使用我们的方法。在第5条中，我们说明了如何扩展这种方法通过增加其他措施。最后一节提出的结论和建议，为今后的工作。

**2 背景及相关工作**

在学术抄袭的诱惑并不仅限于给学生。一名教师涉嫌学术剽窃的情况下说明的粘性问题，举证责任，诽谤，正当程序，以及专业的纪律[3]。努力追求抄袭反诉有一个随之而来的风险，少数担保的决议。然而，我们有一个专业的责任，让我们的学生批判性的思考伦理问题[2]，当它面对抄袭发生[1]。

抄袭检测是一个模式的分析问题。一个抄袭的程序是一个确切的副本原件，或通过运用各种文本转换，如表1所示的一个变种。剽窃检测方法必须出示措施，量化有多近两个程序。除逐字拷贝的情况下，检测方法，使用文本文件的直接比较薄弱，因为没有明显的亲密措施。大多数方法词汇的方法，程序令牌被列为语言关键字和用户符号[4,5,6]。 SIM卡的剽窃检测系统[4]节目源转换成token串，然后比较的使用动态规划字符串对准技术的字符串，如用于DNA字符串匹配。这种功能强大的技术是能够检测示于表1中的程序修改。

**Table 1.** 抄袭修改手段

1. Verbatim copying.

2. Changing comments.

3. Changing white space and formatting.

4. Renaming identifiers.

5. Reordering code blocks.

6. Reordering statements within code blocks.

7. Changing the order of operands/operators in expressions.

8. Changing data types.

9. Adding redundant statements or variables.

10. Replacing control structures with equivalent structures.

YAP家庭的方法[5,6]也分割成源的计划，但只保留表示程序结构 - 控制结构，块，子程序，并使用库函数的令牌。一个语言特定的词汇定义了显著语言元素，包括库函数，标记化。子程序的尸体被扩展一次，他们被称为的顺序。扩大子程序每次后续调用被替换为一个唯一的令牌。由此产生的的令牌字符串是在比较中使用的程序配置文件。计算使用最长公共子序列算法，应用程序配置文件对两种方案的接近程度。 ，YAP方法检测最抄袭转换表1中列出。

**3 我们的方法**

我们的方法创建独立程序大小，数字方案简介。贴近型材的标准化之间的欧几里德距离计算。我们调查的三个配置文件如表2所示：一个集中的物理属性的源文件，另一个在语言使用，第三个结合两种。在这些选择的档案经济建设和表1中给出的检测剽窃变换的鲁棒性。

**3.1 开发环境**

我们使用基于Unix的程序的提交和分级环境。每个学生提供一个私人仓库，提交作业和接收导师回馈。学生环境用于访问存储库包含了一套Unix命令。允许学生无限的意见书，截止控制。所有提交环境维护一个带有时间戳的日志。提交环境有利于捕获的文物用于记录潜在的违规行为，并允许学生工作的历史档案的积累提供了洞察历史学生的行为模式。这个档案还作为学生在这项工作中所使用的程序的测试床。

**3.2 属性度量**

的身体素质P描述了一个程序，在其物理特性：线（L），字（W）和人物（C）的数量。物理配置文件获得UNIX字数（WC）命令。的霍尔斯特德简介?包括的霍尔斯特德度量长度（N），词汇（n）和体积（V）。简介?特点的??程序，在程序令牌的类型和频率。被丢弃。操作令牌对应于源语言的关键字，运算符号，以及标准库模块名称。操作数是程序员定义的字词。

**Table 2.** Physical and Halstead Profiles

|  |  |
| --- | --- |
| Physical | l = line count  w = word count  c = character count |
| Halstead | N = number of token occurrences (Halstead length) n = number of unique tokens (Halstead vocabulary) V = N log2 n ( Halstead volume) |
| Composite | Physical + Halstead |

第三个档案中得到的相结合的物理和赫斯特公司。复合异型材是6个载体（L，W，C，N，N，V）。组合物相结合的相似性检测中的两个或更多的公司的优势。表3示出的类型的转换（来自表1），与物理，Halstead的复合型材是鲁棒的。对于出现复选标记的转换，其中，配置文件将给予相同的值。在其他情况下，则该值可能仅略有不同。请注意，在距离计算中，在一个组件中的小的差异，一般加权向下的其它组分。浅表抄袭转换通常只有小的影响计算出的距离

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Table 3.** Plagiarism Detection Robustness of Various Profiles | | | | | | | | | | |
| Transformation | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| profile P |  |  |  |  |  |  |  |  |  |  |
| profile H |  |  |  |  |  |  |  |  |  |  |
| Composite profile P+ H |  |  |  |  |  |  |  |  |  |  |

**3.3 相关度计算**

两个节目是贴近他们的档案之间的欧几里德距离。相同的程序的紧密度是零。为了建立一个共同的尺度之间比较亲密节目，每个配置文件的标准化，然后计算欧几里德距离。归一化的结果作比较，在一个统一的范围的范围内的零的两个[0,1.414]的平方根。更复杂的计划用于型材和加权曲线组件正常化存在，但没有被调查。计划对排名的亲密值。节目不同的范围和分布的亲密值分配分配。有没有一个先验的亲密阈值表示抄袭。相反，在手从数据确定阈值的位置，利用统计的百分位数。

作为 Unix shell 脚本和 c + + 程序实现的工具来构建和分析这些配置文件。霍尔斯特德分析是最复杂的因为必须标记源应用程序，并且每个令牌列为运算符或操作数。必须构造定义操作员在源语词汇。霍尔斯特德探查器是作为一个 c + + 程序实现的。

**3.4 反抄袭策略**

每个配置文件有其盲点。学生理解体现在配置文件中的措施可以对它们的源代码来误导剽窃监视器进行良性更改。通过添加额外的行的注释，通过拆分或合并物理行，和通过重命名变量，其中一个可以误导的物理配置文件。霍尔斯特德的配置文件，，基于词汇用法，是对这些对策免疫。它是易受影响，但是，对加法永远不会调用的额外代码。它是重要知道盲点的一个配置文件，并克服这些通过使用一个复合的配置文件，使它更难成为学生到掩码剽窃。

亲密关系值的分布取决于编程语言和分配的性质。详细和结构化编程语言 （如 COBOL 导致的程序，看上去非常相像。此类程序的亲密值往往较低级别的频谱的群集。类似的聚类也可能是转让时需要由讲师提供的模块的重用。百分位的方法是确定的程序对其亲密关系会出现例外的首选的方式。

**4 样例输出及观察**

在本节中，我们展示了从在 COBOL 语言中的编程作业的结果。为类收集到的数据是行为的非常反光的课的学生。协作很普遍： 学生倾向于一起工作的两个或更多的 in-groups。协作 （允许和过度） 的亲密关系数据中反映出的模式反映了教师和其他学生的意见。基于这些结果和其他不会在这里报告的经验，我们提出一些意见有关如何解释亲密关系的结果，以及如何建立这"更好的捕鼠器"。

表 4 显示了成对的亲密关系的措施，低于 5 个百分点的三个配置文件。在校学生登录已更换由虚构的名字。第一次观察是，正如预期，亲密关系的一个程序对等级顺序，不同对于每个配置文件，因为每个配置文件的措施不同的特征。每个配置文件可能会显示一组不同的"犯罪嫌疑人"。

|  |  |  |
| --- | --- | --- |
| **Table 4.** Closeness Results -- 5th Percentile | | |
| Physical Profile | Halstead Profile | Composite Profile |
| 0.00000000 alpha alpha  0.00000652 alpha beta  0.00026963 beta gamma  0.00026981 alpha gamma  0.00031262 gamma epsilon  0.00048815 sigma delta  0.00049825 alpha epsilon  0.00050169 beta epsilon  0.00066481 gamma theta  0.00073158 beta theta | 0.00000000 alpha alpha  0.00000000 alpha beta  0.00000000 omega delta  0.00002150 alpha stallion  0.00002150 beta stallion  0.00004025 omega sigma  0.00004025 sigma delta  0.00056763 kappa rabbit  0.00113476 omega rabbit  0.00113476 delta rabbit | 0.00000000 alpha alpha  0.00002785 alpha beta  0.00135498 sigma delta  0.00181437 omega delta  0.00306925 omega sigma  0.00513103 gamma badger  0.00644130 octopus owl  0.00647050 alpha theta  0.00647057 beta theta  0.00758068 delta owl |

第二次的观察是霍尔斯特德配置文件是更多的一项基本措施比物理配置文件。物理配置文件可容纳一个重量测试，侧重于源代码的大小属性。霍尔斯特德配置文件可容纳一个词汇测试，侧重于重要词汇的用法。非内容的文本，如注释和空白，则忽略。因为剽窃的本质就是节目内容，霍尔斯特德亲密关系应作为主要指标和物理贴近作为辅助指标。当基本内容是密切并附带内容也很近时，可能性是较高这些程序是相同的。

第三次的观察是复合配置文件不能合并可预测的方式影响的单独的配置文件。人的十个最小复合配置文件亲密对只有三个是在之间的十个最小的物理内存或霍尔斯特德的配置文件对。十个最大的复合材料亲密关系的配对 （见表 5），只有一人被十个最大的体育和霍尔斯特德之间亲密成对。然而的十个最大体育亲密对 5 人之间的最大的十个复合亲密对。这种现象的解释是亲密关系计算中的每个度量加权的方式。配置文件的最大震级分别与组件具有最高的重量。这有利于物理配置文件中，由于其组件 （例如，字符和单词数） 的值高于霍尔斯特德配置文件组件的值。

剽窃的最强的理由是当亲近 0.000 使用多个配置文件。然而，甚至抄袭的明显例子提出自己，时不明显犯罪行为人是谁，以及是否涉嫌的抄袭是盗窃或相互勾结的结果。显然，这是干预时所需的点。

|  |  |  |
| --- | --- | --- |
| **Table 5.** Closeness Results - Upper Bounds | | |
| Physical Profile | Halstead Profile | Composite Profile |
| 0.02753655 omega queen  0.02777388 badger fox  0.02813216 octopus dove  0.02867729 owl fox  0.02917513 queen elephant  0.02985955 queen rabbit  0.03211501 octopus fox  0.03572043 queen kappa  0.04426516 queen dove  0.04842636 queen fox | 0.05031283 omega dove  0.05031283 delta dove  0.05144719 dove rabbit  0.05163772 dove kappa  0.05325362 dove badger  0.05432432 queen dove  0.05510299 stallion dove  0.05511161 alpha dove  0.05511161 beta dove  0.05544899 dove epsilon | 0.14348834 omega fox  0.14387480 owl fox  0.14745269 queen epsilon  0.14801536 octopus fox  0.15247692 queen dove  0.15793745 queen stallion  0.16328829 queen rabbit  0.17267538 queen elephant  0.17831873 queen kappa  0.24876065 queen fox |

**5 EXTENDING THE BASIC APPROACH**

基本方法是为每个源代码文件，执行成对贴近分析，构建物理，霍尔斯特德，复合型材，然后在某些百分名单公布的亲密。主要嫌疑人是那些谁亲近列表的顶部附近出现。多亲近列表的顶部附近有确凿的证据无异。

在本节中，我们考虑到在正常的分级程序过程中生成的两个二阶产品通过扩展的基本方法。第一个产品是编译错误日志。编译的程序产生一个空的日志 ；否则，生成的错误消息的日志。其他二阶产品是生产程序针对一组标准的测试运行时的执行日志。基于度量比较源代码文件的方法被适用于这些二阶产品，总结在表 6 中。

**Table 6.** Plagiarism Detection on Program Grading Artifacts

|  |  |
| --- | --- |
| Source Code | Construct profiles for each submitted program. Compute source closeness measures.  Identify *source suspects* |
| Compilation  Log | Compile each program, saving compilation log. Construct profiles for each failed compilation. Compute compilation closeness measures. Identify *compilation suspects*.  Compare compilation closeness for source suspects. Refine/Revise compilation suspects list. |
| Execution Log | Execute each program capturing execution log. Construct profiles for each execution log.  Visually inspect execution logs for source suspects. |

剽窃检测为编译日志的重点对异常 — — 通常抄袭是错误维护。物理配置文件是为每个日志而构建的。编译器错误关键字，如"类型不匹配"或"重复定义"词典必须构造来构造的霍尔斯特德配置文件。亲密关系列表生成的编译配置文件和汇编嫌疑人确定。汇编怀疑加强提出的源怀疑的理由。

每个程序执行时，会生成执行日志。构造的物理配置文件。具有相同的行为的程序将具有完全相同的配置文件。因为词汇的程序输出的更改从一个工作分配到另一个任务，必须创建分配特定词汇时，霍尔斯特德使用配置文件。词汇条目反映程序输出要求 （例如，输出标题和菜单和消息的措词）。执行日志亲密关系列表是已被确定为犯罪嫌疑人源的程序生成的。执行怀疑加强源怀疑的理由。

最后一步是以可视方式审查犯罪嫌疑人的确凿证据。基于贴近度列表 （度量） 和目视检查，老师可以适当的干预，哈里斯 [1] 所述。捕获了在整个过程中关于协作模式的亲密关系的证据使教师评估的严重性并采取适当的行动对进攻。显然，应比第一次犯罪者更加严厉地对待个惯犯。

**6 CONCLUSIONS AND FUTURE WORK**

基于度量的物理和霍尔斯特德的配置文件是简单的构造，和它们会导致高效率的计算成对亲密关系的措施。这些度量标准程序提交和分级的上下文中的使用使捕获的学生之间的协作模式的历史记录。干预保证时，这一机构的证据为采取适当的行动提供事实依据。

到目前为止，这种方法已被用来观察，不会追究违法者的协作模式。我们将此技术应用于其他编程的课程，以观察对亲密关系模式分配类型和编程语言的影响。这些研究将提供额外数据，需要了解如何建立亲密关系阈值以下的老师干预有正当理由。我们还将试行过帐匿名亲密的结果，以提高学生对剽窃的认识，激发课堂讨论关于这个问题和其他道德问题的做法。我们将调查哪些配置文件中的组件加权以反映它们的相对重要性替代亲密关系计算。此外，我们会调查由配置文件组件与最大的严重程度 （例如，字符计数的物理配置文件的） 防止统治的归一化公式。最后，我们将正式入便携式系统支持多种编程语言的这种做法。

我们最后的观察，虽然令人不安的是更糟的进攻比抄袭拥有在类的外部的人编写程序学生提交。目前，这种情况可以看出只能通过书面或编程考试。这种困境的解决办法是提交的在时间，或以后，在受控环境中，以识别和删除程序错误注入的自动化错误播种的程序要求的学生。这种方法目前正在调查关于自动化的学生程序 [7] 分级工作结合。

**参考文献**

[1] Harris, J.K. Plagiarism in Computer Science Courses. *Ethics in the Computer*

*Age,* Gatlinburg, TN USA, 1994, 133-134.

[2] Schulze, K.G., and Grodzinsky, F.S. Teaching Ethical Issues in Computer Science: What Worked and What Didn't. 27th SIGCSE Technical Symposium, Philadelphia, PA USA (1996), 98-101.

[3] Kock, N. A Case of Academic Plagiarism. *Communications of the ACM 42,*7 (July 1999), 96-104.

[4] Gitchell, D. and Tran, N. Sim: A Utility for Detecting Similarity in Computer Programs. *Proceedings 30th SIGCSE Technical Symposium*, New Orleans, LA, USA (March 1999), 266-270.

[5] Wise, M.J. Detection of Similarities in Student Programs: YAP’ing may be Preferable to Plague’ing. *Proceedings, 23rd SCGCSE Technical Symposium*, Kansas City, USA. (March 5-6, 1992), 268-271.

[6] Wise, M.J. YAP3: Improved Detection of Similarities in Computer Program and Other Texts. *Proceedings 27th SIGCSE Technical Symposium*, Philadelphia, PA, USA (February 15-17, 1996), 130-134.

[7] Jones, E.L. Grading Student Programs – A Software Testing Approach. *Journal of Computing in Small Colleges 16*, 2 (January 2001), 185-192.