Plagiarism Detection Software

J. Evan Noynaert
Department of Computer Science, Mathematics, and Physics
Missouri Western State College
4525 Downs Drive
St. Joseph, Missouri 64507
noynaert@mwsc.edu

Abstract

Plagiarism is a significant issue on most college and university campuses. Plagiarism detection software is a powerful tool in the fight against plagiarism. However there are serious flaw that instructors must understand. Instructors will rarely receive absolute results from the current generation of software, and manual detective work is usually required to supplement program output.

There are several approaches taken by detection software, although most software relies on comparisons between a document and a large pool of potential sources. There are a variety of algorithms that may be used, but the most popular commercial systems appear to be using RKR-GST based algorithms.

There are many ethical, legal, and practical issues that should be considered in developing the next generation of plagiarism detection programs. Problems with existing programs suggest several features that should be included in next generation software.

The Plagiarism Detection Issue

Plagiarism is a significant problem on almost every college and university campus. The problems of plagiarism go beyond the campus, and have become an issue in industry, journalism, and government activities. Although plagiarism has been a problem for centuries, the Internet and "Copy/Paste" operation makes plagiarism very easy and attractive for students in the twenty-first century.

Some faculty members are not aware of how widespread plagiarism is. Others feel that they can detect plagiarism by careful observation of writing style. Personally, I had always felt that I was good at observing writing styles and I believed that I could detect most cases of plagiarism. I routinely gave zeros for plagiarized work to four or five students every semester. When I tried using plagiarism detection software I was shocked to find that thirty to fifty percent of my student papers contained significant amounts of plagiarized material.

However, plagiarism detection software is not a magic bullet. I found that the detection software produces many false positives. At the same time I was able to detect plagiarized pieces that the software had missed.

At best the current selection of commercial and open source plagiarism detection software is only one tool in the fight against plagiarism. The current generation of tools suffers from significant flaws. Some of the flaws are technical, some flaws are theoretical, and some of the flaws come from a failure to consider ethical and social issues.

Manual Detection of Plagiarism

A brief examination of manual methods of plagiarism detection is useful in understanding how automated detection methods work. Some of the automated detection systems work by mimicking human methods. Other methods depart radically from manual methods.

A second reason for discussing manual methods is that most instructors are going to have to use some of these methods after getting the results of the plagiarism detection software. The results from the detection programs are rarely definitive, and the instructor is going to need to do manual research in order to verify the plagiarism reports.

The first defense against plagiarism involves knowing the body of work from which students are likely to plagiarize. Although it is not possible to know every work, many scholars are familiar with the major published sources.

Another method is the cataloging of past student papers. Some institutions have maintained vaults of student composition papers cross-indexed in several ways. A

teacher who suspected plagiarism could descend into the vault (or more likely, send a teaching assistant) to search for a paper submitted during a previous semester.

Many faculty members detect plagiarism by observing writing styles. Sometimes a paper seems to be too professionally written to have been prepared by a student. Another clue is a sudden shift in writing styles. Ironically, the Copy/Paste process that makes plagiarism so easy also betrays the crime because students forget to reformat the text into a uniform font.

Search engines are often used by faculty members who suspect plagiarism. It is sometimes possible to locate plagiarized work by typing in a few phrases that are suspicious and letting the search engines find matches.

Another method of detecting plagiarism is quizzing students about their written work. A student who has produced their own paper should be familiar with its contents and should be able to answer questions about it. Effective questioning can involve asking students about ideas that were left out or rejected.

All of the manual methods have serious deficiencies. It is impossible to know all of the literature on most topics that undergraduate students are likely to be writing about. The problem is exacerbated by the growing number of informal, unpublished papers available over the Internet. Vaults are limited to papers previously submitted at the institution. The maintenance of the vaults is also labor intensive. Vault systems also depend on a faculty member remembering a similar work that was submitted in a previous term.

Detecting changes in writing styles has several pitfalls. Students often copy from work that was poorly written in the first place, which makes looking for "professional sounding" work ineffective. Some students copy entire works which removes the clue resulting from shifts in writing styles.

Search engines are often ineffective. Some students are quite good at "trivial paraphrasing" which can make searches fruitless. Other problems with search engines result from the original work not being in the search engine's database.

Automated Methods of Plagiarism Detection

There are three different methods of plagiarism detection.

- Quiz methods
- Writing style methods
- Comparison with original sources

The Glatt Plagiarism Screening System involves the quiz method. The program removes words from a student's paper and asks the student to replace the missing words. A score is generated based on the accuracy of the student responses and the amount of time it takes for students to complete the task. (Gaither)

Tripolitis describes a method called "Looking for inconsistency in authorship." The method described by Tripolitis involves statistical comparisons of style in various parts of the document. The method would be useful when the original document is not available. In fact, this method is useful for diagnosing a single paper without having any potential sources available.

Comparison with original sources is the third method of plagiarism detection. There are several products such as CopyCatch Gold that compare student work against a collection of work assembled by the instructor. This is essentially an automated version of the vault system that has been used at some institutions. It seems this method would be particularly effective in composition courses where similar assignments have been made over a period of years.

A variation of the comparison with original sources is comparison with sources available over the Internet. EVE2 is an example of a program that does comparisons across a domain of Internet sites.

Plagiarism.org and its companion Turnitin.com uses comparison against external sources. Turnitin.com saves student papers submitted for analysis in its own internal database. In effect, Turunitin.com has accumulated a vault system that spans all of its client institutions. Turnitin.com also searches various Internet sites including on-line repositories of student papers.

Methods of Comparison

Several methods of comparing text have been suggested for plagiarism detection. Some of these algorithms use basic string matching techniques. Other methods try to construct a "fingerprint" of the documents being compared by searching only for key elements. Unfortunately some of the more promising methods do not appear to be implemented in popular plagiarism detection software.

One of the most well known methods is the Running Karp-Rabin Matching
and Greedy String Tiling (RKR-GST). The algorithm is described by Wise as a method for comparing amino acid biosequences. Despite its origins in biology, the method has application in plagiarism detection. The RKR-GST algorithm appears to be the principle method used in most commercial plagiarism detection software. This algorithm attempts to detect the longest possible string common to both documents.

With the RKR-GST algorithm it is not necessary for the strings to be contiguous in order to be matched. This is a powerful concept because it means that matches can be detected even if some of the text is deleted, or if additional text has been inserted. It is possible for RKR-GST to detect matches even when portions of multiple documents have been combined to create a patchwork of plagiarized material. The algorithm can be further enhanced by parsing the documents to remove trivial words and tokens.

An illustration of the performance of the RKR-GST is shown in the table below and the figure on the following page. The results are from EVE2. Although the software manufacturer does not advertise the algorithms, the results appear to be consistent with what might be produced by the use of an RKR-GST algorithm. In the EVE output duplicated text is shown in red. Italics are used in place of red in the excerpts shown.

Examining the output shows that certain tokens have been eliminated. Punctuation is ignored. The words "of" and "be" and "on" have also been eliminated from the analysis. In the excerpted text the word "you'ld" (sic) was typed by the student as "youd" (sic). In the second excerpt only sporadic phrases are recognized by the algorithm.

The output can look strange to someone who does not understand the RKR-GST algorithm. It appears that some words are being skipped. Even more disconcerting, there are sometimes a few isolated words (such as the "What" in the example) that are scattered through a long block of text that is not identified as plagiarized. These apparent "misses" can undermine the credibility of the analysis if they do not understand the working of the algorithm. The program output is actually showing the attempt of the algorithm to assemble the longest possible strings, even if the strings are not contiguous.

Original text from http://www.jellico.com/spyware.html	Student work
Even if none of *these things happen to you*, you'ld be *surprised at what you might find lurking on your computer*!	*Even if none* of *these things happen to you*, youd be *surprised at what you might find lurking on your computer*!
What is it? Malware (*short for "malicious software*") is any software developed for the purpose of doing harm to a computer	*What* are maiware, adware, and spyware? Maiware is *short for malicious software* and *is* a catch all phrase to describe any software that is designed to damage a computer.

Table 1 Extract from Figure 1, which follows

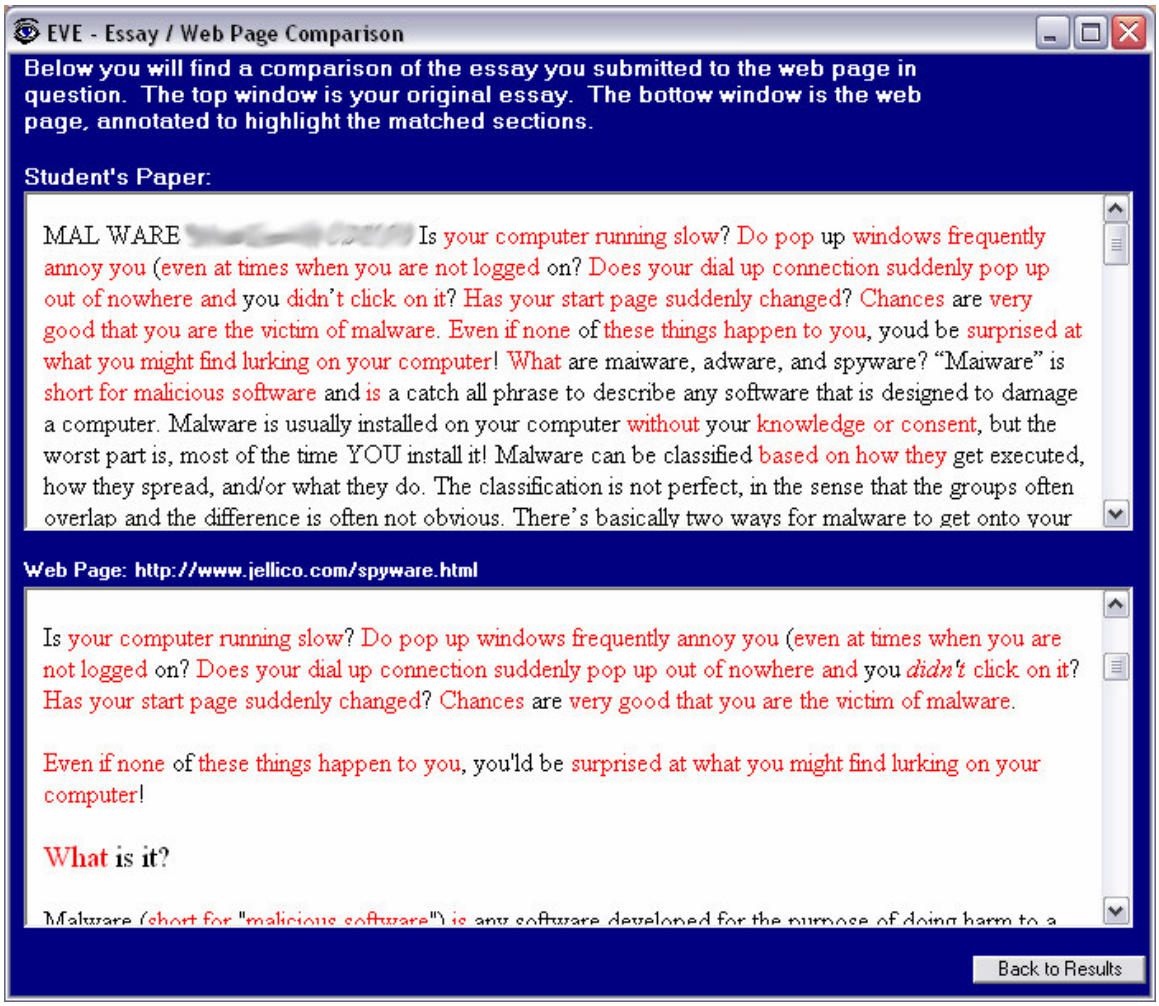

Figure 1, Output from EVE2, The smug represents information which might identify the student.

Other methods of comparison are possible. Several methods based on graph theory have been proposed. Several have been applied to the problem of detecting duplicates in computer programs. In many ways computer languages such as C, Java, and COBOL are easier to analyze than human languages because the syntax and semantics of the computer languages is clearly defined.

Xin Chen *et. al.* have demonstrated a statistical method based on Kolmogorov complexity. To date their method has been applied to detecting duplication in computer programs. However, their methods appear to have potential for evaluating more complex natural language patterns.

Ethical Issues

Ethical issues involved in plagiarism detection are complex, and the discussion here will cover only some of the more obvious issues.

What are the instructor's motives for wanting to do plagiarism detection? I have asked this question on several occasions, and the other person in the conversation is usually perplexed about why the question is even being asked. Plagiarism violates several societal and academic norms. However, I believe that one of the most important aspects of plagiarism detection is teaching students about what plagiarism is, why it is wrong, and how to develop your own thoughts and attitudes. In some cases students do not have confidence in their own writing abilities. Students sometimes feel that plagiarism and rampant paraphrasing is the only techniques that they have available. I have had students tell me that they had high school teachers who told them to write this way. Part of my job as an instructor at an open-enrollment institution is to help students overcome these feelings and learn to write on their own.

Another basic question is "What is plagiarism?" There is general agreement that a word-for-word copy of an entire document is plagiarism. But is "trivial paraphrasing" also plagiarism? I have encountered high school and even college instructors who believe feel that if you change a few words in a sentence the result is not plagiarism. The use of excessive quotations is also a problem. When work is properly quoted and cited, it is probably not plagiarism. However, some students make excessive use of quotations. I have queried students about heavy use of quotations, and I often get a response such as "I couldn't think of a better way to say it."

I believe that from an instructor's perspective the question "What is plagiarism?" does not need to have a definitive answer. I explain to students that even if trivial paraphrasing and excessive quotations are not plagiarism they remain as unacceptable writing styles for college students. I explain that the purpose of writing is not to summarize an article, but rather to synthesize ideas from multiple sources. I also explain situations where quotations are appropriate for dramatic effect or technical accuracy, and that excessive use of quotations is not an acceptable writing style.

Another issue that arises is the ethics of maintaining student work in a vault system, what that system is paper or electronic. Does the work belong to the student, or does it belong to the institution? The question can have legal implications, especially with systems such as Turnitin.com where a third party is holding a copy of the student's work.

It is extremely important that a human make the final determination of whether a work is plagiarized. Faculty members who have not had experience with plagiarism software assume that the software will neatly divide students into two group: those that plagiarized and those who did not. I find that the use of plagiarism detection software usually gives me three groups:

1. The first group contains papers in which no plagiarism was detected by the program or by my reading of the assignment.
2. The second group contains students who clearly plagiarized. Before a paper goes in this pile it is important to locate the source document or documents. Most plagiarism detection software cannot distinguish between properly quoted and cited work and raw plagiarism, so it is important to determine whether there is a

legitimate reason for the duplication. In one famous case a student was accused of plagiarism because his graduate paper had received a very high plagiarism score from a popular plagiarism detection program. Further investigation showed that student had posted a version of his paper on a web site, and the detection software had found a high degree of similarity between the web posting and the final document. (Dehnart) This demonstrates the importance of carefully examining every report.

3. The third group are the gray cases. There are usually substantial number of papers in this pile. It is not uncommon to spend an hour investigating a single paper. In many cases I am never able to resolve whether plagiarism occurred, or I remain convinced that it probably occurred, but I cannot find the original source. The quality of the program output can be very helpful at this point. For example, the output from Turinitin.com is fairly helpful in identifying the source, but EVE2 often gives a long list of possible plagiarized sources which must be laboriously checked.

A Next Generation of Plagiarism Detection Software

Each of the current programs for plagiarism detection has serious deficiencies. Some deficiencies will never be overcome. However, there are several steps that could be taken to improve the next generation of plagiarism detection software.

Ethical, Legal, and Practical Issues

Ethical, legal, and practical issues should be considered at the beginning of the development process, and they should be an integral part of program development. For many of these questions there is no absolute correct answer, so it is important to allow the user to select options that match institutional and personal preferences. One example is the question of ownership of the student paper. Users should have a choice about whether the student work should be retained, and if it is retained whether it should be uploaded to a global vault or remain in a local vault controlled by the institution. The program should also allow the user to set standards for plagiarism and select among various definitions of plagiarism.

Algorithm Selection

More theoretical and practical work needs to be done on the subject of detection algorithms. Although the RKR-GST algorithm has proven to be effective in many situations, it still has several weaknesses.

Most research effort seems to be directed at detecting plagiarism among a large number of potential matches. There is also a need for algorithms that specialize in confirming duplication once a potential match has been found by other methods.

An ideal plagiarism detection system would use more than one algorithm for detection of duplicate work.

There should also be work on systems that check for proper quotation and citation.

Presentation

There should be greater care and attention given to presenting the results of a detection run.

Most of the current programs label all duplicates as plagiarism, although most also have disclaimers that say that the results should be confirmed. It would be better if software reported some type of "Duplication Index" rather than the currently popular "Plagiarism Index." The determination of whether plagiarism occurred should be made by a human, not a computer program.

The software should make it easy to trace the results back to the source documents. Although most current programs make an effort to help identify original sources it is often difficult to find exact correspondences needed to prove a charge of plagiarism.

Openness

The person using the plagiarism detection software should be able to determine the algorithms being used, and there should be a way to trace back the exact reason that a particular result was reported.

There are two arguments about why the algorithms remain secret. One argument is that the precise algorithm is a trade secret of the company. Another argument is that if students knew the algorithms they might be able to figure out how to plagiarize in a manner which would fool the detection software.

The trade secret argument is only significant if the software is produced by a commercial enterprise. If the software is produced in an open source environment the need for maintaining trade secrets evaporates. However, even in a commercial environment the person using the software should be aware of the strengths and weaknesses of the algorithms being used in the detection process.

Making the algorithms open might make it possible for students to defeat the detection system. However, if multiple detection algorithms are used in the program it would be much more difficult to find a method that would defeat all the methods in the system. In addition, the majority of people who plagiarize are doing so as a shortcut. Attempting to defeat the plagiarism detection algorithm would be a great deal of extra work. Besides, there is already one established method to avoid detection: don't plagiarize.

Steps in the development of a next generation plagiarism detection system

The following points should be considered in the development of a next generation of plagiarism detection software.

1. Identify ethical, legal, and practical issues at the first stages of the development process, and make sure that these issues are considered at each stage of the development process.
2. Carefully consider the model for system development. An "Open Source" development model would allow for a high level of transparency.
3. Consider the algorithms and programming language. Most of the existing systems have been developed in C, C++, Java, or similar languages. Development in an Artificial Intelligence oriented language such as LISP or using Artificial Intelligence tools such as AIXML should be considered.
4. Consider alternate sources for document comparisons. The user should be able to select to search for matches on the Internet or from local and global document depositories. It should be possible to perform searches against local databases without exposing the document to off-site networks. This last issue can be particularly important to businesses and government agencies.
5. The user interface should be easy to use. Many of the existing systems suffer from significant use interface problems and poor programming techniques that make the programs frustrating to use. The existing systems also tend to be very slow; it may take a day or more to complete an analysis of even a modest sized class. Efficient algorithms and efficient coding techniques should be a priority.

References

Denhart, Andy, "The Web's Plagiarism Police", June 14, 1999, http://www.salon.com/tech/feature/1999/06/14/plagiarism/print.html.

Gaither, Renoir, "Plagiarism Detection Services", http://www.lib.umich.edu/acadintegrity/instructors/violations/detection.htm, Last updated November 30, 2004.

NetLink 2000, "Malware: spyware, adware, viruses, and trojans", http://www.jellico.com/spyware.html.

Schleimer, Saul, Daniel S. Wilerson, Alex Aiken, "Local Algorithms for Document Fingerprinting", International Conference on Management of Data, 2003, ACM.

Tripolitis, Konstantinos, "Automatic Detection of Plagiarism", August 30, 2002, http://www.dcs.shef.ac.uk/teaching/eproj/msc2002/pdf/m1kt.pdf, page 10

Wise, Michael., “Running Karp-Rabin Matching and Greedy String Tiling”, Technical Report Number 463, The University of Sydney, March 1993, http://www.it.usyd.edu.au/research/tr/tr463.pdf.