

**HERMETIC AND WEB PLAGIARISM DETECTION  
SYSTEMS FOR STUDENT ESSAYS—  
AN EVALUATION OF THE STATE-OF-THE-ART**

**TUOMO KAKKONEN**

**MAXIM MOZGOVOY**

*University of Joensuu, Finland*

**ABSTRACT**

Plagiarism has become a serious problem in education, and several plagiarism detection systems have been developed for dealing with this problem. This study provides an empirical evaluation of eight plagiarism detection systems for student essays. We present a categorical hierarchy of the most common types of plagiarism that are encountered in student texts. Our purpose-built test set contains texts in which instances of several commonly utilized plagiaristic techniques have been embedded. While Sherlock was clearly the overall best hermetic detection system, SafeAssignment performed best in detecting web plagiarism. Turnitin was found to be the most advanced system for detecting semi-automatic forms of plagiarism such as the substitution of Cyrillic equivalents for certain characters or the insertion of fake whitespaces. The survey indicates that none of the systems are capable of reliably detecting plagiarism from both local and Internet sources while at the same time being able to identify the technical tricks that plagiarizers use to conceal plagiarism.

**INTRODUCTION**

Park (2003) defines plagiarism as “the theft of words or ideas, beyond what would normally be regarded as general knowledge” (p. 472). Plagiarism has become a serious problem in education because of, among other factors, the widespread



access that students who use computers have to the resources of Internet. The term *student plagiarism* is often used to refer to the incidences of plagiarism committed by students who attend educational institutions. Posner (2007) recently estimated that one-third of all high-school and college students have committed some kind of plagiarism. The results obtained from several surveys of plagiarism among students make Posner's estimation seem overly optimistic. Park (2003) has listed the following most common reasons why students commit plagiarism: ignorance about the correct method of using citations and references, a desire to obtain better grades by incorporating superior material without proper acknowledgment, an inability to manage schedules (inadequate time management skills), and the perception that the risk of getting caught is remote.

The most common method of detecting plagiarism relies on the ability of a teacher to make deductions about the probability of plagiarism on the basis of internal clues embedded in the text itself. Thus, for example, a student whose style is habitually laconic and pedestrian might suddenly become uncharacteristically eloquent in patches. Countering plagiarism by using such "traditional" means is unfortunately ineffective since few assessors, especially in an era of information explosion, can be sufficiently familiar with all the sources of a topic to identify precisely the origin and extent of a delinquent student's plagiarisms. In such traditional settings, an assessor can only confirm an initial suspicion of plagiarism by manually matching the suspected plagiarisms to the original sources—a task which, if at all possible, is time-consuming and frustrating.

Teachers and academics abhor plagiarism because it is inconsistent with pedagogical aims. The mere copying of texts has no conceivable educational value. Moreover, it involves students in moral compromise and deception, and is not fair to the honest students that do not plagiarize. The ease with which anyone can copy and collate text from the Internet makes web plagiarism deeply tempting to some students. A number of researchers, such as Lathrop and Foss (2000), suggest that the use of the Internet has significantly increased the amount of plagiarism that students commit.

While the ready availability of a great variety of information on the Internet has arguably increased the incidence of plagiarism, the Web has also indirectly provided educators with various means of countering it. While Internet search engines, such as Google, can be used to detect Internet plagiarism, the detection process is, by any standards, both tedious and labor-intensive because it requires an assessor to insert parenthesized extracts from a suspected text into the search engine. It is obvious that such a process is extremely time-consuming.

While academic interest in plagiarism detection in software source codes dates back to the 1970s (Ottenstein, 1976), plagiarism detection systems for texts have only been developed over the past two decades. In this article, we introduce some of the most prominent of existing text plagiarism detection systems in conjunction with the results of a study which we undertook to evaluate their ability to detect types of plagiarism ranging from direct copying to paraphrasing.

### Automatic Plagiarism Detection

The most common method used by plagiarism detection systems involves the use of string matching algorithms that enable the comparison of document content (for example, Joy & Luck, 1999; Wise, 1996). In contrast to *hermetic detection systems* that scrutinize only local collections of documents, *web plagiarism detection systems* attempt to identify instances of plagiarism outside of the set of essays that is being analyzed. Some systems are capable of performing both types of detection. The fact that a system is capable of web detection does not necessarily imply that such a system has to be used online. Some web plagiarism detectors are installed on the instructor's own computer.

Web plagiarism detection is typically based on using an existing web search service, such as Yahoo or Google, for searching the content of the Internet for documents matching to a student's written composition. In contrast to most web plagiarism detection systems that are sold commercially, most hermetic detectors are available without cost (Bloomfield, 2009; Prechelt, Malpohl, & Philippsen, 2002; Schleimer, Wilkerson, & Aiken, 2003), are also, in some cases, open source (Ahtiainen, Surakka, & Rahikainen, 2006; Joy & Luck, 1999) and utilize detection algorithms, the details of which have already been published in academic publications. This allows a knowledgeable user not only to apply a certain system as a black-box, but also to make certain assumptions about its capabilities on the basis of the techniques that the system utilizes.

Most of the current hermetic systems compare input files on the basis of the content (although other approaches, such as attribute counting, are also used; Faidhi & Robinson, 1987). The most popular algorithm is probably a variation of the well-known string matching algorithm Running Karp-Rabin Greedy String Tiling (RKR-GST), a method that has been implemented in a number of systems such as JPlag (Prechelt et al., 2002), Plaggie (Ahtiainen et al., 2006), and YAP3 (Wise, 1996).

### Requirements for a Successful Plagiarism Detector

The purpose of this section is to establish a set of requirements against which the evaluated systems will be compared in the evaluations that will follow. An automatic plagiarism detector has to be able to identify two kinds of plagiarism:

1. the plagiarism involved in copying from another student's work; and
2. the plagiarism involved in copying without acknowledgment from reference materials (such as those in textbooks and the Internet).

In addition to the source of the plagiarized texts, there are, in fact, numerous styles of plagiarism. These range from direct copying to making use of a ghostwriter. A detection system should ideally be able to detect all instances and styles of plagiarism while refraining from the stigmatization of texts in which no plagiarism has in fact occurred.

Because automatic plagiarism detection systems have to be able to identify several kinds of cheating, their text comparison routines can be rather complex. We have devised the following hierarchy of the most common types of plagiarism that we will use throughout the present article to characterize the types of plagiarism and plagiarism detection methods. Some of the issues were discussed earlier (Maurer, Kappe, & Zaka, 2006).

1. Verbatim copying
  - a. Copy-paste copying from an electronic source.
  - b. Word-for-word transcription of texts from a non-electronic source.
2. Hiding the instances of plagiarism by paraphrasing
  - a. Adding or removing words or characters.
  - b. Adding deliberate spelling and grammar mistakes.
  - c. Inserting words with similar meanings (synonyms).
  - d. Reordering sentences and phrases.
  - e. Effecting changes to grammar and style.
3. Technical tricks exploiting weaknesses of current automatic plagiarism detection systems
  - a. The insertion of similar-looking characters from foreign alphabets. Thus, for example, the letter "O" can be equally well represented with the following three different characters: Unicode 004F (Latin O), 039F (Greek Omicron), and 041E (Cyrillic O).
  - b. The insertion of invisible white-colored letters into what seem to be blank spaces. Most modern text processors allow the user to specify a font color in a document. The plagiarism could exploit this feature by inserting a white font in a blank space with a white background. This would have the effect of distorting the content of the text even though, to the naked eye, it would be visually identical to the original.
  - c. The insertion of scanned text pages as images into a document.
4. Deliberate inaccurate use of references
  - a. The improper and inaccurate use of quotation marks: the failure to identify cited text with the necessary accuracy.
  - b. The deliberately inaccurate use of bogus references: the making of references to incorrect or non-existent sources.
  - c. The use of "forgotten" or expired links to sources: the addition of quotations or parentheses but a failure to provide information or up-to-date links to the sources.
5. "Tough plagiarism," the types of plagiarism that are particularly difficult to detect for both humans and computers
  - a. The plagiarism of ideas: the use of similar concepts or opinions outside the realm of common knowledge without due acknowledgment.
  - b. The plagiarism of translated text: translations unsupported by acknowledgment of the original work.

- c. The production of text produced by an independent “ghostwriter.”
- d. Artistic plagiarism: the presentation of someone else’s work in a different medium (the end result may involve text, images, voice or video).

Some of these methods of plagiarizing are easier to detect than others. An instance of plagiarism resulting from “verbatim copying” (type 1) might be detected with the simplest string matching routine. An example of “paraphrasing plagiarism” (type 2) may require the use of Natural Language Processing (NLP) techniques such as morphological analysis and syntactic parsing. Type 3 consists of tricks that some students that know the way in which detection systems work, are deliberately using in order to fool the systems. We expected, based on our knowledge on the methods used in state-of-the-art plagiarism detectors, the existing systems to be limited to detecting these three types of plagiarism (and, as our results below show, with some serious limitations).

The detection of type 4 (deliberate inaccurate use of references) is not yet something that plagiarism detection software are capable of finding. This would call for using sophisticated automatic reference and citation tracking tools, something that is as yet unavailable. We refer to the types of plagiarism listed in item 5 as tough plagiarism. These plagiarism types are extremely difficult to detect and finding them automatically is beyond state-of-the-art and most likely will remain so within the foreseeable future.

In addition to being capable of accurately detecting the types of plagiarism listed above, a detection system should avoid making false detections (i.e., not return false positives). Since the volume of texts on the Internet is so vast, it is possible for parts of a student’s text coincidentally to resemble text from an existing web page—even though the student might never have seen the web page concerned. This kind of resemblance is often referred to as “casual similarity.” Students may also cite materials that they themselves published on the Internet. It is, of course, perfectly legal to quote fragments from informally published self-authored text.

### **Evaluation of Plagiarism Detection Systems**

There have been several reviews of plagiarism detection systems conducted previous to this study (e.g., Clough, 2000; Lancaster & Culwin, 2004). However, these studies did not report explicitly on experimental evaluations of the accuracy and false detection rates of the detection systems concerned. Instead, they described the various shortcomings and advantages of the systems without offering empirical evidence to back up their claims. The systematic survey of plagiarism detection systems that was undertaken by Maurer et al. (2006) is one of the rare exceptions. These researchers compared the capabilities of the following three popular products in this field: Turnitin, SA, and *Docol©c* ([www.docoloc.com](http://www.docoloc.com)). The evaluations of Maurer et al. (2006) revealed the following limitations: None of the three systems were able to . . .

1. detect paraphrasing techniques such as the substitution of words with synonyms, the use of alternative syntactic forms for the same expression of content, and the rewording of phrases.
2. identify sources that were not already available online.
3. detect instances of plagiarism across languages.

It has been suggested that detecting plagiarized sources not already available online is becoming less and less of a problem as more and more sources are made available electronically. Maurer et al. (2006) also argue that instances of the third type of plagiarism listed above will be difficult, if not impossible, to detect in the foreseeable future. They therefore argue that the best results that can currently be achieved from automatic detection will be obtained from systems are able to detect plagiarism that is heavily reliant on paraphrasing.

Some of the plagiarism detection research (e.g., Maurer et al., 2006; Whale, 1990) is dedicated to the evaluation methods and measures for plagiarism detection systems. Whale (1990), for instance, explains how one can adapt *recall* and *precision*—the standard evaluation measures in Information Retrieval (IR)—for evaluating plagiarism detection systems. This method was also applied by Verco and Wise (1997). Hoad and Zobel (2003) have suggested the use of *the highest false match* (the highest percentage given to an incorrect result) and *separation* (the difference between the lowest correct result and the HFM) as a metrics for determining the quality of a plagiarism detector. Research by Mozgovoy et al. (2005) has also shown how “a jury” that consists of several high-quality plagiarism detection systems can be used to test another system with good results and a high degree of reliability.

## EMPIRICAL EVALUATION OF SELECTED PLAGIARISM DETECTION SYSTEMS

### The Systems Evaluated

Four of the eight detection systems selected for this study are commercial products and the remaining four are either products of academic research or free services. The systems were selected based on their availability and the fact that they represent different types of approaches to automatic plagiarism detection. One of the commercial systems we wanted to evaluate could not be included in the study because the company that developed it refused to allow us to include its system in a comparative evaluation study. We were unable to include another system, namely Plagiarismdetect.com, in the evaluation, because it produced too many coincidental matches and false detections to be reliably evaluated.

In the following three subsections, we introduce the plagiarism detection systems included in this study. The systems are divided into three categories, namely to those that can perform hermetic or web detection and those that are capable of both types of detection.

### Hermetic Detection Systems

*WCopyfind* (WCF; version 2.6; Bloomfield, 2009): This system is a downloadable software tool that is designed to compare natural language texts. “This program examines a collection of document files. It extracts the text portions of those documents and looks through them for matching words in phrases of a specified minimum length. When it finds two files that share enough words in those phrases, WCF generates HTML report files. These reports contain the document text with the matching phrases underlined” (Bloomfield, 2009). WCopyfind contains few settings that allow users to fine-tune the detection algorithm (these include, for example, “ignore numbers,” “ignore letter case,” and “skip non-words”). The detection process is simple: the user can add several documents to the system at once, and start the comparison process. In our experiments, we used the default settings of the system. The system accepts plain text and MS Word \*.doc files.

*Sherlock* (version 3.2.2; Joy & Luck, 1999): This plagiarism detection system was implemented as a part of the BOSS electronic course management instrument at the University of Warwick, UK (Heng, Joy, Boyatt, & Griffiths, 2005). Sherlock allows one to compare both natural language-based and source code files. Because the system has been written in Java, it requires the installation of a Java Runtime Environment. The system allows the user to add a number of plain text documents to a local collection before beginning analysis. Results are reported not only in the form of the usual tables, but also in the form of graphs of matches that depict individual documents as vertices and matching pairs as edges.

*AntiPlagiarist* (AP; version 2.0): This system, which was developed by ACNP Software, is a stand-alone plagiarism detection tool that offers a functionality that is very similar to that of WCopyFind. Although this system has been designed for the commercial market, a free evaluation version is available. AP supports numerous document formats, including MS Word and Word Perfect files, HTML, plain text, and Rich Text Format documents. The user has the option of indicating the particular files that need to be tested for plagiarism, and can even identify particular documents that might be suspect. The system has only one fine-tunable parameter that affects the detection process: the number of words that must be matched (we used the default value of 16 in our experiments). The system prints pairs of similar documents when it detects the presence of instances of plagiarisms.

### Web Detection Systems

*SeeSources.com* (SS) is a free web-based service that allows a user either to copy/paste or to upload a file (in MS Word, HTML, or ASCII format) for the purposes of plagiarism detection. No settings are offered to the user. The search takes place in two phases: *source search* and *deep search*. We considered that

QA:  
WCopyFind  
or  
WCopyFind P

the system had been successful in detecting plagiarism if the deep search results produced instances of plagiarism.

The *EVE2* (Essay Verification Engine) system, developed by Canexus, keeps no database of its own essays or texts. Its method is to search the Internet for essays by making use of existing web searching engines. We used version 2.5 of this system in our research. The program installs to the user's own computer and allows the user to select from three possible detection modes: *quick*, *medium*, and *full strength*. Adding a file to the system is a cumbersome process because it only accepts up to ten files at once.

*Plagiarism-Finder* (PF; version 1.3.0; Mediaphor) works in more or less the same way as *EVE2*. It installs onto the user's computer and searches the Internet for possible occurrences of text fragments from the local document collection. PF allows a user to adjust the detection algorithm by making adjustments to two parameters: the record length (in words) that needs to be checked, and the increment (in words) that defines the size of the step whereby advances are made to the next "record" (a sequence of words) in the document. While the selection of a small record length and increment usually results in more matches, it also slows down the detection process and can produce *false detections* (see Section 3).

#### *Systems that Perform Both Web and Hermetic Detection*

*SafeAssignment* (SA), one of *MyDropBox*'s tools (Sciworth Inc., 2009), is a system that can perform both hermetic and web detection. While the user interface is intuitive, the addition of new essays for scrutiny is complicated by the fact that these essays need to be submitted one by one. The system gives the user no control over the detection method that should be used.

*Turnitin* (IParadigms, 2008), which claims that thousands of schools and universities around the world use its services, is arguably the most widely used of all current plagiarism detection systems. The system produces an originality report on a student text by comparing it not only to the pages and documents on the Internet, but also to its own essay database of more than 40 million student papers.

### **Method**

The main purpose of this study was to evaluate the extent to which the existing systems are able to detect plagiarism of the type 1 (verbatim copying), 2 (paraphrasing), and 3 (technical tricks) that we expected to be within the capabilities of the state-of-the-art systems. We seek answers to the following research questions:

- How accurate are the state-of-the-art plagiarism detection systems in detecting plagiarism ranging from verbatim copying and paraphrasing to the use of technical tricks?
- How prone are the evaluated systems to making false detections?

We performed our evaluations by first collecting a set of test documents from several sources (Internet, books, paper mills) that contained several types of plagiarism. We then performed three types of tests to evaluate the accuracy and false detection rate of the selected plagiarism detection systems in web and hermetic detection as well as in detecting type 3 of plagiarism (technical tricks). The following subsections provide detailed descriptions of the test data and evaluation procedures we applied.

### *Test Data*

The construction of the test set consisted of three phases. In the first phase, we collected a set of sentences. The sentences were sourced from texts in the following fields: the use of technology in education, natural languages, natural language parsing and understanding, automatic essay grading, artificial intelligence, and object-oriented programming. The sentences belonged to the following three categories: Original, Web, and Mill. The sentences in the *Original* category were either deliberately written for the test set by the authors, or were sourced from books that had not already been published online. These sentences were aimed at observing the false detection rate of the systems. The *Web* sentences were obtained from a variety of Internet pages. The papers in the *Mill* category were acquired from “paper mill” services. Paper mills, such as 123HelpMe.com (2009) and all free essays.com (2009), are Internet services that offer students ready or custom-made essays. These two sets of sentences were aimed at evaluating the accuracy of detection.

The test sentences described in the previous paragraph constituted the *verbatim sentences* to which no edit operations were applied (type 1, verbatim copying). We then modified these sentences in three different ways to simulate the type 2 (paraphrasing) plagiarism. As the result, we had three groups of sentences that were edited, synonymous, or paraphrased. The *edited* sentences were characterized by minor alterations such as added spaces, intentional spelling errors, deleted or added commas, and periods that were replaced by exclamation marks. In the *synonymous* sentences, one or two words from each sentence were replaced with exact or close synonyms. The *paraphrased* sentences were characterized by a wide range of sentence alterations of the following kind: they included the kind of alterations found in the edited and synonymous sentences, and, in addition, changes in the original order of words and phrases. Thus, for example, the sentence, “He drank water and beer and she ate chocolate,” could be rearranged so that it read, “She ate chocolate and he drank beer and water.”

This test set contained a total of 1,200 sentences: 100 sentences for each test sentence type. In the last phase, we divided the sentences into test files, with each file containing only sentences that represented a single test type. We also preserved the original order of the sentences so that each sequence of test sentences formed a coherent passage of text.

The list of possible methods for hiding plagiarism that was set out above contained three specific techniques from type 3 (technical tricks) plagiarism, which can be applied semi-automatically with very little or no effort on the part of the plagiarizer. These techniques consist of: (a) using similar-looking characters of different alphabets; (b) inserting white-colored font into empty spaces; and (c) inserting parts of scanned text as images. In order to evaluate a detection system's ability to detect obfuscations of type 3, we created an additional set of test documents that we called *Tricks*, which helped us to gauge the extent to which various systems are able to detect this kind of cheating. We created this test data in the following way. We took all the verbatim documents from each category (Original, Web, and Mill) and created three obfuscated versions that contained *fake characters*, *fake spaces*, and *scanned text* from each of these 12 files. Each of these characteristics has the following meanings:

- *Fake characters*: Some of the characters in the document are replaced by similar-looking counterparts from other alphabets. Thus, for example, the word "computer" can be spelled with Cyrillic c, o, p, and e—Unicode characters 0x0441, 0x043E, 0x0440, and 0x0435.
- *Fake spaces*: All the white spaces in the document are replaced with white-colored letters.
- *Scanned text*: The body text of the document is replaced by its scanned version which is inserted into the document as an image file.

This resulted in the 84 test files described in Table 1.

#### Procedure

QA:  
EVE  
or  
EVE 2?

*Web detection*: Five of the seven detection systems—SafeAssignment, SeeSources, Turnitin, EVE, and Plagiarism-Finder—are capable of web plagiarism detection. Hence, we selected these systems for the web detection evaluation. For each system, we carried out separate test runs for the files that belonged to each of the three test categories (Original, Web, Mill). We carried out these evaluations in the following way: To begin with, we inserted all the texts in the test set into the detection systems that were to be evaluated. Next, after running the detection procedure, we observed the number of documents in each test category (Original, Web, Mill) and edit type (verbatim, edited, synonymous, paraphrased) that the system being evaluated reported as having alarming levels of plagiarism. Finally, we calculated the evaluation scores using our two evaluation metrics, *precision* and *recall* (see the definitions of these measures in Subsection "Evaluation measures" below).

The tests were first run with the default settings of each of the detection systems. For those systems which permit the user to exercise control over the detection mechanism, we ran the tests on the most stringent settings available. We refer to these settings as "strict." Table 2 summarizes the settings we used for each system.

Table 1. The Test Set

Test type	Edit type	No. of files	No. of sources	Topics
Original	Verbatim	4	4	Natural language understanding and parsing, artificial intelligence
	Edited	4		
	Synonymous	4		
	Paraphrased	4		
Web	Verbatim	4	6	Automatic essay assessment, parsing, artificial intelligence, object-oriented programming
	Edited	4		
	Synonymous	4		
	Paraphrased	4		
Mill	Verbatim	4	8	Technology in education, natural languages, robotics, virtual reality
	Edited	4		
	Synonymous	4		
	Paraphrased	4		
Tricks	Fake characters	12	18	All the above topics
	Fake spaces	12		
	Scanned text	12		
TOTAL		84	18	

**Note:** The column "Number of files" lists the number of files in each of the test categories. The column "Number of sources" indicates the number of sources used in the construction of the sentences. The column "Topics" lists the topics of the test sentences found in the tests files.

*Hermetic detection:* Five of the eight detection systems are capable of hermetic plagiarism detection. These are AntiPlagiarist, SafeAssignment, Sherlock, Turnitin, and WCopyfind. In hermetic detection, no matches can be made to online documents. The only matches that are made are to those documents that are contained in a system's internal database. The most obvious way to evaluate this kind of system is therefore to check whether it is able to match modified files (edited, synonymous, paraphrased) to their original versions (verbatim).

We carried out these evaluations in the following way: To begin with, we inserted all the verbatim texts (representing original, unmodified texts) from Original, Web, and Mill test sets into the detection systems that were to be evaluated. It should be noted that the Original, Web, and Mill collections in this test should be treated merely as three different sets of documents. Their origin cannot be discovered by a hermetic plagiarism detection system. While a system

Table 2. The Test Settings for Each of the Systems

System	Default	Strict
SafeAssignment	Default	—
SeeSources	Default	—
Turnitin	Default	—
EVE2	Normal	Full strength
Plagiarism-Finder	Normal, record length 7 words, increment 50 words	Detailed, record length 4 words, increment 2 words

**Notes:** The columns “Default” and “Strict” describe the default and strictness of the test settings for each of the systems respectively.

could thus identify the fact that a certain file A is a modified version of a certain file B, it cannot know whether B is an original essay or merely a document obtained from a paper mill. We then carried out separate test runs for the remaining files (edited, synonymous, and paraphrased) that belonged to each of the test categories (Original, Web, Mill). Each test run therefore consisted of 12 files. Finally, we calculated the evaluation scores using the two evaluation metrics, precision and recall.

*Revealing Methods of Hiding Plagiarism by Using Technical Tricks:* We used the Tricks test data to evaluate the extent to which the systems are capable of handling technical tricks in hiding plagiarism (type 3). This evaluation could be for each of the eight systems. We expected that any reasonable plagiarism detection system would be able to recognize all such modifications and either identify the obfuscated file as a match of the original document or that it would at least be able to alert a user of the system to abnormalities in the student essay.

#### Evaluation Measures

We adopted the approach suggested by Whale (1990), and used *recall* and *precision* for reporting the evaluation results. While recall is used as the measure of the accuracy of detection, precision measures the false detection rate. The use of recall and precision for evaluating plagiarism detection accuracy can be defined as follows:

$$recall = \frac{\text{detected instances of plagiarism}}{\text{total instances of plagiarism}} \quad (1)$$

$$precision = \frac{\text{detected instances of plagiarism}}{\text{detected instances of plagiarism} + \text{false detections}} \quad (2)$$

We define these measures on the document level. Thus, an example of a plagiarized document is any document that contains one or more plagiarized passages. For example, a test document collection that contains four documents out of which two have at least one instance of plagiarism, would receive the recall score 0.5 in our evaluation scheme.

## Results and Discussion

### *Web Detection*

The Original category enabled us to observe the precision of the systems. This test indicates how prone the evaluated systems are to producing false detections while performing web plagiarism detection. None of the systems (apart from SeeSources) marked any of the files in the Original category as plagiarized. The precision score of SeeSources on that data set was 0.82. SeeSources was therefore the only systems that returned false detections.

Table 3 presents the results of the web detection experiment for Web and Mill categories. As the documents in these two tests consisted of texts copied from the Internet, they did not contain plagiarism-free documents (i.e., possible false detections). The only way in which a false detection could occur in the Web and Mill categories would be if a detection system claimed that file X had been plagiarized from file Y, while file X was in reality using file Z as the source of plagiarism. This did not occur in our experiment. Only the recall figures are therefore reported in the table.

In Table 3, a recall score of 1.0 indicates that the system had perfect detection accuracy (i.e. it was capable of detecting each of the files in the test set to contain plagiarism). A recall score of 0.0, by contrast, indicates that the system did not find any of the files to contain alarming levels of plagiarism. The figures on the “default” column give the recall score on the default settings of each of the detection systems. For those systems which permit the user to exercise control over the detection mechanism, the column “strict” gives the results for the recall scores on the most stringent settings available (see Table 2).

SeeSources scored the highest average recall, and was only unable to detect instances of plagiarism in the Mill paraphrased test set. The most obvious defect of this system was the number of false detections it generated on the Original data (as discussed above, it was the only system in the experiment to generate false detections). Hence, we can conclude that while SeeSources is capable of finding most instances of plagiarism, it does so by permitting some false detections to occur.

We judged SafeAssignment to be the best system overall in terms of this evaluation. In contrast to SeeSources, it did not generate any false positives. It scored 100% in the Original and Web test sets. In the Original texts, SafeAssignment identified between one and three sentences in some of the

Table 3. Web Plagiarism Detection Evaluation Results

	SS	SA	TurnitIn	EVE		Plagiarism-Finder	
	Default	Default	Default	Default	Strict	Default	Strict
<b>Web</b>							
Verbatim	1.00	1.00	0.75	0.00	0.00	0.00	0.00
Edited	1.00	1.00	0.75	0.00	0.00	0.00	0.00
Synonymous	1.00	1.00	0.75	0.00	0.00	0.00	0.00
Paraphrased	1.00	1.00	0.50	0.00	0.00	0.00	0.00
AVERAGE	1.00	1.00	0.69	0.00	0.00	0.00	0.00
<b>Mill</b>							
Verbatim	1.00	1.00	0.50	0.00	0.00	0.00	0.00
Edited	1.00	1.00	0.75	0.00	0.00	0.00	0.00
Synonymous	1.00	0.00	0.50	0.00	0.00	0.00	0.00
Paraphrased	0.50	0.50	0.25	0.00	0.00	0.00	0.00
AVERAGE	0.88	0.63	0.50	0.00	0.00	0.00	0.00

**Key:** SS = SeeSources, SA = SafeAssignment

documents as sentences that had been copied from a web page. But because the overall plagiarism score (between 3 and 11%) for the test document in each of these cases was smaller than the 25% limit set for the “yellow” category with a moderate plagiarism risk, we did not classify these cases as false detections. The detection accuracy of SafeAssignment was also excellent in the Web category. It returned a plagiarism score of less than 100% for only two of the 16 files in this category. One of these was a file that was embedded with synonyms, and the other was a file that had been primed with paraphrased sentences.

The performance of TurnitIn permitted it to be rated between the top (SafeAssignment and SeeSources) and low (EVE and Plagiarism-Finder) performers. Even though TurnitIn did not produce any false detections from the Original data, its detection accuracy was poorer than that of SafeAssignment in all test categories except for the synonymous tests on the Mill data. TurnitIn found a few coincidental matches in one document in all the Original test categories. But because the system did not classify the document as a whole to be exhibiting an alarming level of plagiarism, we did not accept these matches as instances of false detections. While testing the Web texts, TurnitIn failed to detect a document that had been sourced from a web page on the .fi domain. It therefore failed to produce accurate results for that document in all the test categories. This lapse is indicative of the existence of a gap in the comprehensiveness of TurnitIn’s Internet search coverage. The percentage of files that the system identified as plagiarized on the Mill data was higher than the figures given

in Table 3. In some cases, the source from which we had taken our test sentences was not among the sources indicated in Turnitin's output. Since our test requirement was the correct identification of each instance of plagiarism as well as its source, we did not accept these files as correctly identified.

Although EVE returned no false identifications for the Original sentences on either the default or the strict setting, its overall performance was nevertheless deplorable on both settings because of its lack of detection accuracy. It was not, for example, able to identify a *single* instance of plagiarism on our test set! In order to be certain that this performance had not been caused by the file format or the test settings, we carried out the detection runs with all three of the possible detection levels offered by the tool and repeated the experiment with both Plain Text and Microsoft Word documents, both of which EVE claims to support. Even so, its performance did not improve. In order to make sure that our installation file had not been corrupted, we sent an inquiry to the company that developed the tool and explained the detection problems that we were encountering. We did not receive a reply.

Plagiarism-Finder failed to impress us with its default settings, and its performance was generally similar to that of EVE. While Plagiarism-Finder did not generate any false alarms with the Original test data, it was only able to identify low overlaps with Internet sources—even with the Web verbatim documents. But the overlapping sections had, in most cases, not been obtained from the documents from which we had sourced the sentences. For one of the Web verbatim files (in which the whole document consisted of sentences that had been copied from the Internet), the system identified only 9% of the sampled words as having been sourced from the Internet. It also identified a total of 1% of the essay as directly copied!

This example demonstrates Plagiarism-Finder's inability to detect even verbatim plagiarism. The most common result of the system in this evaluation was a set of three word fragments that matched different Internet sources. A closer inspection of the results produced by Plagiarism-Finder provide even more causes for concern. Almost 100% of the matches in the Web category did not link to the sources from which we had taken the sentences. We also noticed that almost all the detections were made in text from web pages in the .de, .at, and .ch domains or in links to the Excite.de search engine. We are therefore left to conclude that the system only searches a small German section of the Internet and that it uses a web search engine with rather poor coverage. Such problems alone would make the results (when using the strictest settings) very difficult to interpret—to say the least—were it not for the fact that they make the system useless in practice.

#### *Hermetic Detection*

In the hermetic detection evaluation, we observed which of the modified files (edited, synonymous, paraphrased) were found to be plagiarized from the

Table 4. The Recall Figures for the Hermetic Detection Experiment

Method	Dataset	Sherlock	WCF	AP	Turnitin	SA
Edited	Original	1.00	1.00	1.00	0.00	0.00
	Web	1.00	1.00	1.00	1.00	0.25
	Mill	1.00	1.00	0.75	1.00	0.50
	AVERAGE	1.00	1.00	0.92	0.67	0.25
Synonymous	Original	1.00	1.00	0.75	0.00	0.25
	Web	1.00	1.00	1.00	0.75	1.00
	Mill	1.00	1.00	0.75	1.00	0.25
	AVERAGE	1.00	1.00	0.83	0.58	0.50
Paraphrased	Original	1.00	1.00	0.25	0.00	0.00
	Web	1.00	1.00	0.75	0.50	0.50
	Mill	1.00	0.67	0.75	1.00	0.00
	AVERAGE	1.00	0.89	0.58	0.50	0.167
AVERAGE		1.00	0.96	0.78	0.58	0.31

**Key:** WCF = WCopyfind, AP = AntiPlagiarist, SA = SafeAssignment

**Notes:** Because the method of hiding the instances of plagiarism is more important for the evaluation of hermetic detection than is the source of the test data, we used a slightly different way of representing the results than those shown in Table 3.

corresponding verbatim texts. Table 4 presents the results of the experiment by giving the recall figures for each system. As all of the test cases in this experiment contained plagiarism (and hence there was no possibility of false detections), only the recall figures are reported in the table.

In Table 4, a recall score of 1.0 indicates that the system had perfect detection accuracy (i.e., it was capable of detecting each of the files in the test set to contain instances of plagiarism). For example, Sherlock detected that all the edited documents in the Original dataset contained plagiarized sections from their corresponding verbatim files in the Original dataset. Hence, the system received the recall score 1.0 the Original edited test. In contrast, Turnitin detected only half of the documents in the Web paraphrased test set of containing plagiarizes sections. Hence, it received the recall score 0.5.

Sherlock performed best in the hermetic detection test. It was indeed able to identify the fact that all the test documents contained alarming levels of plagiarism. WCopyfind was the second-best performer. It missed only a few paraphrased documents in the Mill test set. It is interesting to note that the two systems that performed well in the web plagiarism test (SafeAssignment and

TurnitIn) were less successful in the hermetic test, scoring an average of 0.31 and 0.58, respectively. It appears that these two universal (web and hermetic) detectors use web-based detection to reveal hermetic plagiarism. This can be achieved by applying the following scheme. First, a system finds external (web) sources that match the analyzed files of a local collection. Then a pair of files is marked as plagiarized if both files match the same external source. This means that in those cases where a universal system does not find any matches to external sources, as was the case with the Original data, it fails to detect hermetic plagiarism.

Although its performance was not among the best, TurnitIn behaved in the way we expected. As the complexity of the modifications gradually increased from edited to paraphrased, the accuracy of the system's detections decreased. We were somewhat surprised to find that SafeAssignment achieved its highest accuracy score in the Synonymous dataset. AntiPlagiarist performed somewhere between the two highest (Sherlock and WCopyfind) and lowest (TurnitIn and SafeAssignment) ranking groups and achieved an average recall of 0.78. This system also behaved in the way we expected it to as its performance dropped gradually from the edited to the paraphrased test set.

#### *Technical Tricks Plagiarism*

The results of the evaluation of the type 3 (technical tricks) plagiarism are summarized in Table 5. As none of the systems were able to detect any instances of plagiarism from the documents in this experiment (the Tricks test set), it would not make sense to report recall or precision scores. The table hence describes the behavior of each system when faced with documents containing specific types of technical tricks.

None of the evaluated systems were able to detect any instances of plagiarism from the documents in this experiment. TurnitIn was the only system that performed in the desired way and warned the user about the potential problems in the documents. TurnitIn also asks the user to confirm that the uploaded document is the correct one by displaying it in its final format before starting the detection process (as a document converted from MS Word, PDF, or other supported formats to ASCII text). It was interesting to note that TurnitIn also produced different error messages for each of our test categories. These messages are designed to alert the user to potential problems in the file.

When SafeAssignment scans documents with Cyrillic and fake space data, there is a chance that the user will accept that a document with character codes (as replacements for the fake spaces and the Cyrillic letters) has a plagiarism score of 0 if he or she does not first open the document for closer inspection. The three other commercial systems—AntiPlagiarist, EVE, and Plagiarism-Finder—failed to detect anything suspicious in the Trick documents. They all ran the detection procedures without any warnings and so failed to find any instances of plagiarism.

Table 5. The Results of the Technical Tricks Evaluation

System	Method	Response
<b>Web and Hermetic Detection Systems</b>		
SA	Cyrillic	The system inserted character codes to replace Cyrillic letters. The detection process ran normally.
	Fake space	The detection process ran normally.
	Scanned text	The system refused to process.
	Conclusion	Although the detection process ran without any warnings in the presence of Cyrillic and fake space data, a closer inspection of the document revealed the attempt to hide plagiarism by the character substitutions. But since no warnings were given to the user, it is possible that the semi-automatic plagiarism tricks were unnoticed by the teacher.
Turnitin	Cyrillic Fake space Scanned text	An error message was produced and the detection did not run.
	Conclusion	The system refused to process any of the files. But it alerted the user to the fact that there was something suspicious about the document(s) in question.
<b>Web Detection Systems</b>		
SS	Cyrillic	The system accepted the files and found no instances of plagiarism. Some files were not analyzed because of the error message: "Not enough text for analysis. Please try again!"
	Fake space	The system accepts the files and finds no instances of plagiarism.
	Scanned text	The system processes an empty document but does not warn the user.
	Conclusion	The system runs the detection without explicitly warning the user (see comments below).
EVE	Cyrillic Fake space Scanned text	While the system accepts the files, it finds no instances of plagiarism and does not alert the user in any way.
	Conclusion	This system has not been designed to detect semi-automatic plagiarism-hiding methods.

Table 5. (Cont'd.)

System	Method	Response
PF	Cyrillic	The system replaces the Cyrillic letters with some Latin Extended Unicode characters (such as ñ, î, á) and runs the detection normally.
	Fake space	The system replaces fake spaces with the original letters that were set in font color white in MS Word. It runs the detection normally but without any warnings.
	Scanned text	Although system processes an empty document, it does not warn the user.
	Conclusion	Because the system has been designed without an awareness of semi-automatic plagiarism hiding methods, it runs the detection without any warnings.
<b>Hermetic Detection Systems</b>		
AP	Cyrillic Fake space Scanned text	Although the system accepts the files, it finds no instances of plagiarism and does not alert the user in any way.
	Conclusion	The system has been designed without an ability to detect semi-automatic plagiarism hiding methods.
Sherlock	Cyrillic	Although system accepts the files, it finds no instances of plagiarism and does not alert the user.
	Fake space Scanned text	It is impossible to use this system in these cases because Sherlock only accepts plain text files.
	Conclusion	This system has not been designed to detect the "Cyrillic letters" trick. Other kinds of simple plagiarism-hiding methods also do not work because of the limitations and restrictions of the system.
WCF	Cyrillic Fake space Scanned text	Although the system accepts the files, it identifies no instances of plagiarism and does not alert the user in any way.
	Conclusion	The system has been designed without an ability to detect semi-automatic plagiarism hiding methods.

Even though SeeSources does not give the user any warning message, it is possible to detect that there are problems on the screen that one uses for launching the detection process. Scanned documents, for example, will appear as empty and the Cyrillic characters will be replaced with non-ASCII symbols. A skilled and well-informed user will therefore be alerted to the presence of tricks.

The two research systems, Sherlock and WTF, seem to concentrate on the detection algorithms. They both ignore the possibility that the detector can be deceived during the very first step by the insertion of incorrect input data. This distinctive feature is caused by the fact that these plagiarism detectors are research-oriented “proof-of-concept systems” rather than production-ready software products. Sherlock was the only system that did not accept input files in MS Word or PDF formats, a peculiarity that made the running of fake space and scanned tests impossible.

## CONCLUSIONS

Considering the limitations of modern methods of computerized plagiarism detection, one should not underestimate the usefulness and effectiveness of techniques of effective manual analysis. A number of guidelines in this regard had been issued, for example, by Jackson and Kern (2003) and Harris (2004). Some of the advice that they offer includes the following:

- Take a closer look at examples of peculiar or unwarranted formatting, such as the unnecessary insertion of line breaks or the sudden appearance of text blocks in fonts that are unnecessarily different from that in the main text.
- Analyze quotations. These may be outdated (often an indication of “reusing” an old paper with even older citations) or inconsistent.
- Check the student’s writing style for consistency with the earlier works produced by the same student. It is also grounds for suspicion when adjacent paragraphs are written in radically dissimilar styles even when they are still dealing with the same topic. Other inconsistencies in texts should also alert the instructor to the possibility of plagiarism.

The importance of plagiarism *prevention* cannot be overstated. A vast variety of successful techniques that include the use of creative assignments (to which it is difficult to find the ready solutions), penalties, honor codes, better student-teacher interaction, etc., have all been reported in the literature. While the authors of this article acknowledge the importance of these measures, they fall outside the scope and ambit of this research and hence will not be considered here in any more detail. The best sources of further information on these non-computerized methods are Lathrop and Foss (2000), Park (2003), and Posner (2007).

While automatic or computer-aided plagiarism detection is not the final and definitive solution to all the problems involved in student plagiarism, they can provide teachers with much needed help in enforcing the anti-plagiarism policies

of educational institutions. In order to provide an overview of the most advanced available methods in automated detection, we have offered empirical evaluations of the current state of automatic plagiarism detection in student essays, and made various suggestions about what still needs to be done in order to make further progress in this field.

### **Results of the Empirical Evaluations**

While the results of our web detection experiments revealed that although SafeAssignment was the detector that offered the best all-round performance, the only category in which it failed to produce a perfect performance was in the paper mill test set. This seems to indicate that the internal essay database of SafeAssignment might be insufficiently comprehensive. While the free SeeSources system returned the highest degree of detection accuracy in the experiment, it also returned false positives. The other commercially available Internet-based service in our evaluation, Turnitin, performed reasonably well. Turnitin's main advantage at this stage is that it continually augments and makes use of the vast database of essays that it constantly collects from users of the system. The performance of EVE2 and Plagiarism-Finder can only be described as catastrophic because both of these systems failed to detect a single instance of plagiarism from our test set.

As one might expect, it was synonymous and paraphrased test documents that caused the most trouble for the web detection systems. This is exemplified the average score of 0.125 of the four systems in the web plagiarism test on the Mill synonymous test set.

In contrast to what happened in the case of the web detection evaluations, none of the systems in hermetic evaluation failed completely. The best performers in this category were Sherlock and WCopyfind. While Sherlock demonstrated a perfect recall for all of the tests, WCopyfind missed only a few paraphrased documents in the Mill test set, and scored an average recall of 0.96. In contrast also to what happened in the web plagiarism experiments, the edited category was not much easier for the hermetic systems to handle than were the other two categories (synonymous and paraphrased). In fact, the average recall of 0.77 achieved on synonymous data was better than the 0.73 achieved on the edited test set. One of the most surprising findings was the poor performance of Turnitin when confronted by the Original test set. This indicates that Turnitin is unable to identify intersections between two local files if they are not presented by an online-published source. We also found it surprising that the other Internet-based commercial system (SafeAssignment) scored an average recall as low as 0.31, and that it failed to produce impressive results in any of the test categories.

One should also bear in mind that most systems have not made any provision to detect simple technical tricks such as the substitution of Latin letters with their non-Latin counterparts or the insertion of fragments of scanned documents into

text. Turnitin was in fact the only system that performed satisfactorily in this category. Such methods of concealing plagiarism are very simple to perform and equally simple to detect. We therefore are of the opinion that any current system should take care to include appropriate methods for the detection of such modes of deception.

We wish to state in conclusion that the most advanced methods of detection make it possible reliably to detect the kind of verbatim copying and plagiarism that is produced by means of minor editorial changes to sources obtained both from the Internet and locally. It is unfortunate, however, that these two basic requirements were not offered by any of the systems that we tested. Our test data makes it clear that the various means of web or hermetic detection offered by each of these systems is therefore far from perfect or even adequate. Turnitin and SafeAssignment, the only two systems that are capable of both web and hermetic detection, both exhibited considerable deficiencies. SafeAssignment, which performed best with instances of web detection, produced poor results when it was required to perform hermetic evaluation. This compelled us to give it the lowest possible ranking of all the five systems. Its handling of the type 3 (technical tricks) plagiarism was also not without problems.

While Turnitin, by contrast, was not the best system when it came to web or hermetic evaluation, it could nevertheless be ranked in the middle of the scale with regard to both types of detection. It was also able to warn the user about the possible existence of plagiarism "tricks." One of Turnitin's greatest advantages (that gives it preeminence in certain areas over the other systems) is that it is able to scrutinize the considerable collection of essays that it has sourced from its clients and made part of its database. Our test set generated plagiarism warnings by examining these essays, especially in terms of the Mill data.

All these observations lead us to the conclusion that none of the systems that we have tested is able to provide a complete solution to all the problems of plagiarism detection and that much more research therefore needs to be conducted in the field of automatic plagiarism detection methods.

### **Future Perspectives**

Since most of the existing plagiarism detection systems do not perform any significant NLP, Uzuner, Katz, and Nahsen (2005) and Mozgovoy, Kakkonen, and Sutinen (2007) are among the few who have incorporated NLP techniques in the service of plagiarism detection, they are consequently only capable of detecting direct copying and the kind of plagiarism that consists of simple editorial changes such as the addition, the removal, or substitution of various characters.

We believe that the most significant problems that remain to be solved in the field of plagiarism detection are those that deal with authorship attribution, the tracking of references and citations, "fingerprinting," and the use of NLP and content comparison techniques. Since it is unrealistic to assume that all possible

sources of plagiarism will be contained in the detection system's internal collection or in public Web access, the attribution of authorship by means of fingerprinting methods is of crucial importance. The possibility of detecting suspicious stylistic variations within a single text or between two separate texts produced by the same author is essential because it indicates the strong probability of plagiarism. This capability, in combination with the use of NLP and IR techniques, will help researchers to produce more advanced versions of plagiarism detection software for identifying plagiarism of types 1 (verbatim copying), 2, and 3. Despite the existence of apparent challenges, it appears to be feasible to modify existing reference and citation tracking methods so that they will be able to overcome the problems presented by the frequently non-standard and inconsistent methods of the referencing used by students. This is the basis on which to develop methods of detecting type 4 (deliberate inaccurate use of references) plagiarism.

So long as the methods that had been described above are not yet available, techniques that involve rewording, paraphrasing, false citations, and translation plagiarism still remain powerful tools for deception in the hands of plagiarizers. It is unreasonable to expect successful progress in the detection of tough plagiarism while aspirant plagiarizers can still use rewording, paraphrasing, and fake citations with impunity for circumventing the detection methods of the various systems that are currently being used. We, therefore, strongly believe that current research efforts in plagiarism detection should be directed toward resolving the remaining problems in detecting the "simpler" types of plagiarism (types 1, 2, 3, and 4) before researchers make concerted attempts to tackle the problems inherent in the detection of type 5 plagiarism.

Current systems are also unable to detect instances of plagiarism that consists of non-textual data such as images, music, or website design. At the present time, one can only speculate as to how widespread these forms of plagiarism might be among students in the field of arts and design.

## REFERENCES

- 123HelpMe.com. Retrieved September 29, 2009, from <http://www.123helpme.com>
- ACNP Software. *AntiPlagiarist*. Retrieved September 29, 2009, from <http://www.anticutandpaste.com/antiplagiarist/>
- Ahtainen, A., Surakka, S., & Rahikainen, M. (2006). Plaggie: GNU-Licensed Source Code Plagiarism Detection Engine for Java Exercises. *Proceedings of the 6th Baltic Sea Conference on Computing Education Research, Koli, Finland*.
- All free Essays.com. Retrieved February 5, 2009, from <http://www.allfreeessays.net>
- Bloomfield, L. A. (2009). *Software to Detect Plagiarism: WCopyfind (Version 2.6)*. <http://www.plagiarism.phys.virginia.edu/Wsoftware.html> (Retrieved September 29, 2009).
- Canexus Inc. *EVE2- Essay Verification Engine*. Retrieved February 5, 2009, from <http://www.canexus.com/>

- Clough, P. (2000). *Plagiarism in natural and programming languages: An overview of current tools and technologies*. Internal Report CS-00-05, University of Sheffield, UK. Docol©c. Retrieved September 29, 2009, from <http://www.docoloc.com>
- Excite.de. Retrieved September 29, 2009, from <http://www.excite.de>
- Faidhi, J., & Robinson, S. (1987). An empirical approach for detecting program similarity within a university programming environment. *Computers & Education*, 11(1), 11-19.
- Harris, R. (2004). Anti-plagiarism strategies for research papers (dated November 17, 2004). Retrieved September 29, 2009, from <http://www.virtualsalt.com/antiplag.htm>
- Heng, P., Joy, M., Boyatt, R., & Griffiths, N. (2005). *Evaluation of the BOSS Online Submission and Assessment System*. Research Report CS-RR-415, University of Warwick.
- Hoad, T. C., & Zobel, J. (2003). Methods for identifying versioned and plagiarised documents. *Journal of the American Society for Information Science and Technology*, 54(3), 203-215.
- iParadigms. (2008). *Turnitin.com. Digital assessment suite*. Retrieved September 29, 2009, from <http://turnitin.com>
- Jackson, R., & Kern, K. (2003). *Deterring and detecting plagiarism. Instruction commons guides*. E-Library @ Iowa State University. Retrieved March 3, 2008, from <http://www.lib.iastate.edu/commons/resources/facultyguides/plagiarism/deter.html>
- Joy, M., & Luck, M. (1999). Plagiarism in programming assignments. *IEEE Transactions on Education*, 42(2), 129-133.
- Lancaster, T., & Culwin, F. (2004). Using freely available tools to produce a partially automated plagiarism detection process. *Proceedings of the 21st ASCILITE Conference, Perth, Australia*.
- Lathrop, A., & Foss, K. (2000). *Student cheating and plagiarism in the Internet era. A wake-up call*. Englewood, CO: Libraries Unlimited.
- Maurer, H., Kappe, F., & Zaka B. (2006). Plagiarism—A survey. *Journal of Universal Computer Science*, 12(8), 1050-1083.
- Mediaphor Software Entertainment AG. *Plagiarism-Finder*. Retrieved September 29, 2009, from <http://www.m4-software.com/>
- Mozgovoy, M., Fredriksson, K., White, D., Joy, M., & Sutinen, E. (2005). Fast plagiarism detection system. *Lecture Notes in Computer Science*, 3772, 267-270.
- Mozgovoy, M., Kakkonen, T., & Sutinen, E. (2007). Using natural language parsers in plagiarism detection. *Proceedings of SLATE'07 Workshop, Farmington, Pennsylvania*.
- Ottenstein, K. (1976). An algorithmic approach to the detection and prevention of plagiarism. *ACM SIGCSE Bulletin*, 8(4), 30-41.
- Park, C. (2003). In other (people's) words: Plagiarism by university students—Literature and lessons. *Assessment & Evaluation in Higher Education*, 28(5), 471-488.
- Posner, R. A. (2007). *The little book of plagiarism*. New York: Pantheon Books.
- Prechelt, L., Malpohl, G., & Philippsen, M. (2002). Finding plagiarisms among a set of programs with JPlag. *Journal of Universal Computer Science*, 8(11), 1016-1038.
- Schleimer, S., Wilkerson, D., & Aiken, A. (2003). Winnowing: Local algorithms for document fingerprinting. *Proceeding of the International Conference of the Association for Computer Machinery Special Interest Group on Management of Data*. San Diego, California.
- Sciworth Inc. (2009). *MyDropBox*. Retrieved February 5, 2009, from <http://www.mydropbox.com/>

- SeeSources.com—Instant, automatic & free text analysis. Retrieved September 22, 2009, from <http://seesources.com/>
- Uzuner, Ö., Katz, B., & Nahnsen, T. (2005). Using syntactic information to identify plagiarism. *Proceedings of the Association for Computational Linguistics Workshop on Educational Applications*. Ann Arbor, Michigan.
- Verco, K., & Wise, M. (1997). Plagiarism à la mode: A comparison of automated systems for detecting suspected plagiarism. *The Computer Journal*, 39(9), 741-750.
- Whale, G. (1990). Identification of program similarity in large populations. *The Computer Journal*, 33(2), 140-146.
- Wise, M. J. (1996). YAP3: Improved detection of similarities in computer program and other texts. *Proceedings of the 27th SIGCSE Technical Symposium on Computer Science Education*. Philadelphia, Pennsylvania.

Direct reprint requests to:

Tuomo Kakkonen  
Department of Computer Science and Statistics  
University of Joensuu  
P.O. Box 111  
FI-80101 Joensuu, Finland  
e-mail: [tuomo.kakkonen@cs.joensuu.fi](mailto:tuomo.kakkonen@cs.joensuu.fi)