SNITCH: A Software Tool for Detecting Cut and Paste Plagiarism

Sebastian Niezgoda and Thomas P. Way
Applied Computing Technology Laboratory
Department of Computing Sciences
Villanova University
Villanova, PA 19085

sebastian.niezgoda@villanova.edu
thomas.way@villanova.edu

ABSTRACT

Plagiarism of material from the Internet is a widespread and growing problem. Computer science students, and those in other science and engineering courses, can sometimes get away with a "cut and paste" approach to assembling a paper in part because the expected style of technical writing is less expositional than in liberal arts courses. Detection of cut and paste plagiarism is time-consuming when done by hand, and can be greatly aided by automated software tools. This paper reports on the design of a software tool called SNITCH that implements a fast and accurate plagiarism detection algorithm using the Google Web API. Issues related to plagiarism detection software are discussed and empirical results of a performance and accuracy study are presented.

Categories and Subject Descriptors

K.4.1 [**Computing Milieux**]: COMPUTERS AND EDUCATION – *Public Policy Issues.* H.3.3 [**Information Systems**]: INFORMATION STORAGE AND RETRIEVAL – *Information Search and Retrieval.*

General Terms

Algorithms, Design, Management.

Keywords

Plagiarism detection, ethics, cut and paste plagiarism, automated grading tools, cheating

1. INTRODUCTION

Internet plagiarism is a significant problem, with a recent study finding that 40% of students admit to having used a "cut and paste" approach in at least one writing assignment, while 77% do not feel that such cheating is serious [7]. Plagiarism is certainly nothing new, and many strategies have been developed to deal with the problem. Educational institutions at all levels employ a comprehensive approach to dealing with plagiarism which involves setting a policy against it, judiciously enforcing the policy, actively encouraging students to avoid cheating, designing plagiarism-proof assignments, and using whatever tools and techniques are available to detect instances of cheating. This approach can be effective, and with a recent resurgence in the successful use of honor codes [7] and the use of plagiarism detection software [5], many forms of cheating can be reduced.

The problem of plagiarism in student papers and reports written for computer science, and other science and engineering courses, is fundamentally different than in expositional and narrative writing more common in non-technical courses. The style of a technical paper can be more disjoint, reflecting more of an assembly of assertions rather than a holistic theme, and still be appropriate to the subject matter, if not ideal. Such a terse style lends itself to cut and paste plagiarism while an expositional style requires much more effort to use such an approach. Thus, the more flowing a paper is, the more likely any plagiarism used would be of a larger scale, such as purchasing a paper outright from an online database or term-paper mill [6,9].

Manual approaches to detecting plagiarism are labor intensive, involving multiple readings of each suspect document while relying on the expertise of the reader to recognize instances of plagiarism. This can involve detecting stylistic differences between the author's expected voice and the voice expressed in the paper, use of unfamiliar or unexpected terminology, and recognizing verbatim text from an outside source [6]. The use of software is ideally suited to automating the detection of verbatim plagiarism, but is not capable of higher-levels of detection. However, with the availability of material from the Internet, and the prevalence of cut and paste plagiarism, software can serve as a valuable tool for catching, or deterring, the cheater. [5]

This paper describes the design of an algorithm for automated plagiarism detection and an associated software tool called SNITCH (Spotting and Neutralizing Internet Theft by CHeaters) that implements the algorithm. The SNITCH program uses a sliding window to scan a document and locate candidate passages that might be plagiarized. Each passage is searched for on the Internet, and an annotated HTML report is output containing the original document with hypertext links inserted for any passages that were found in an Internet search. A brief summary, including statistics about the amount of plagiarism, if any, and the time taken to perform the search, is also provided in the report.

SIGCSE'06, March 1–5, 2006, Houston, Texas, USA.

2. PLAGIARISM DETECTION

Detecting plagiarism can be a tedious, time-consuming and repetitive task, all characteristics that make the problem ideally suited to a software solution. In this section, a brief discussion of the common, non-software-based approach to plagiarism detection is provided, followed by an overview of existing software solutions and issues to be considered in the design of such software solutions.

2.1 The Manual Approach

In a time with a seemingly limitless electronic cache of material from which to "borrow" from the Internet, an approach commonly used by educators to combat the cut and paste approach to plagiarism is to highlight suspicious excerpts in a paper, and then enter them into an online search engine. If identical excerpts are found in an online source, it is likely that the excerpt was plagiarized. Certainly, if multiple such instances of identical excerpts are discovered in a single paper, a strong case can be made that intentional plagiarism is present.

The downside of this manual approach is that it is labor intensive, requiring detailed, on-screen reading and re-reading of each paper, coupled with the repeated use a search engine including copying and pasting selected passages from each paper using a mouse and keyboard. Although this tedious approach is perhaps less exhausting than leafing through textbooks looking for potential matches, or being intimately familiar with enough such textbooks and other sources to recognize stolen phrases at a glance, it is nevertheless daunting, particularly when faced with a large stack of student papers to evaluate under deadline pressure.

2.2 Existing Software

Software has been developed that reduces a lot of the labor-intensive aspects of cut and paste plagiarism detection. There are a number of commercially available software tools and services that perform automated checking of student papers. Often the cost is relatively high, or the turn-around time is sufficiently long, or both, reducing the availability to those educators with budget constraints. These available automated approaches often assume that large sections of a paper, or even entire papers, would be copied verbatim. Yet, for technical oriented research papers, such as in computer science and engineering disciplines, a cut and paste approach where paragraphs, sentences or even phrases can be gathered into a report is easier to get away with. Where there is less narrative, this more fragmented approach is easier to pass off as finished writing and also harder to recognize as plagiarism.

Software tools have been successfully used for detection of plagiarism in student programming assignments for many years [4,5,10]. Two program files are compared after some compiler-like preprocessing to try to find similarities in the files that could indicate plagiarism. This form of software continues to prove an invaluable tool both for the detection of cheating and for grading assistance in large section courses [10].

In the author's experience, the Eve2 software performs adequately for shorter papers, but its report generation feature can be inaccurate. Although Eve2 is really designed to determine how closely a student paper matches a single online source, for simply detecting the simple presence of plagiarism it is acceptable. It costs $29.95 for an unlimited use license, and takes from 2 to 45 minutes to scan a typical 5-7 page paper, depending on the thoroughness of the desired scan. [2]

TurnItIn is a well-respected service, where student papers are submitted via the Internet for analysis. Reports are generated and returned to the instructor, normally within four to six hours of submission. The service can be expensive, with yearly institutional subscriptions available ($3,000/year) or a license fee and per-student charge ($530/year plus $1/student) among the options. [12]

The MyDropBox service is a recent and able competitor to TurnItIn, with a similar pricing strategy and turn-around time. Reports are generated within 24 hours of submission. A number of pricing plans are available, such as an institutional plan that costs approximately 60 cents per student. [8]

An extensive survey was conducted at Claremont-McKenna College to measure the efficacy of all available plagiarism detection software. For detecting cut and paste plagiarism, the results overwhelmingly favored the use of TurnItIn, with Eve2 and manual Google searches combined with WCopyFind the only other worthwhile alternatives (at the time of the study). [5]

2.3 Issues

There is some debate about whether the use of plagiarism-detection software can injure the relationship between educator and student. Although issues of mutual trust are important, when software is used judiciously, it can encourage learning, reduce cheating, increase fairness for all students, and redirect the burden of proof of plagiarism from the instructor to the software. [6] Aside from these philosophical and psychological issues, there are a number of technical issues that should be considered and addressed when crafting such a tool:

Identification – A human would look through the document for vocabulary and phrasing that seemed out of place for the particular author. To attempt to duplicate this behavior, the software could use a simplified semantic analysis, looking for sequences of word of some configurable maximum length. These sequences can then be ranked based on metrics such as a count of sufficiently long words, or average length of words in the sequence, as indicators of more advanced writing. The justification for this approach is intuitive; technical writing tends to be dense and terminology-rich, leading to a higher than average word length. It is more difficult for a non-expert in a domain (i.e., a student) to rapidly prepare a report of seeming significance without resorting to cut and paste plagiarism to get the job done.

Thoroughness – Because identifying plagiarism is time-consuming, an instructor with a large stack of papers to grade may disqualify a paper as soon as any instance is detected. The software approach can measure the degree of plagiarism, and automatically provide written documentation of the cheating in much less time. The software should be configurable to determine how many candidate passages should be searched for, and provide a way for disqualifying a paper when a certain number of verified instances of plagiarism have been detected. This limiting enables thorough analysis of papers to be performed while providing an upper-bound on time, which are essential considerations when faced with a lot of work to do in a short period of time.

Flexibility – An instructor can recognize an "interleaved" instance of plagiarism, where some copied material has been slightly revised by replacing, adding or deleting one or more words to avoid detection. Software can duplicate this approach, although it is a difficult combinatorial problem to solve. Because search engines allow for "wildcards" (e.g., the insertion of a '*' to represent any sequence of letters or words), a simple approach of replacing any short or common words in a candidate passage with a wildcard may be an effective technique to combat attempts to defeat detection of cut and paste plagiarism. Using wildcards can increase the possibility of false positive results, so their use means extra time must be spent inspecting the results.

Arms Race – Any software tool that is available for use by instructors could also be used by students, with the concern being that students may try to defeat automated plagiarism detection by using such a program while writing their papers. Recognizing that plagiarism can never be completely eliminated, any techniques that cause a potential cheater to read, revise and analyze written material, is an improvement over the alternative. Rather than the educator and student involved in a plagiarism detection arms-race of sorts, such software may lead to improved learning. As a side-effect of trying to out-engineer the plagiarism software, perhaps the student cheater may wind up inadvertently understanding the subject matter of the paper quite well.

The issues and approaches raised here, and the features and techniques used in previous tools and manual methods, provide the motivation for the design and implementation of the SNITCH software.

3. DESIGN OF SNITCH

This section describes the design of the plagiarism analysis and detection algorithm used in SNITCH, and provides implementation details that support the algorithm, including integration of the freely available Java-based Google Search Application Program Interface (API).

3.1 Algorithm

The algorithm developed for SNITCH uses a sliding window technique and average length per word metric to identify potential instances of plagiarism. In general, the algorithm uses the following steps:

- Open a document
- Analyze the document
  - Read a window containing the first/next **W** words
  - Measure the number of characters for each word
  - Calculate the **Weight** of the window, the average number of characters per word for the words in the window
  - Associate this **Weight** with this particular window for use later
  - Repeat for all such windows in the document, shifting the window forward in the document by 1 word
- Search for plagiarized passages
  - Order windows in decreasing order, and eliminate overlapping windows
  - Rank all windows in decreasing order by **Weight**
  - Select the top **N** weighted windows, and search the Internet for each, gathering the top search result (if any) for each
- Generate a report – Create an HTML document containing statistics of search time, number of searches performed, percentage of document found to be plagiarized, and other pertinent statistics. Include the original document with embedded HTML tags linking plagiarized passages to their sources on the Internet.

The algorithm is parameterized to allow variation of the size of the sliding window (W) and number of searches performed (N), to enable fine-tuning on a per-user basis. Decreasing W will lead to more potential candidates, but may increase false positive results because the fewer words there are in a search phrase, the more likely they could occur by chance. Increasing W can improve the confidence in individual search results, but if set too high, it may reduce that likelihood that any matches will be found if the window is larger that the plagiarized passage. Increasing or decreasing N will increase or decrease the thoroughness, and lengthen or shorten the time taken to analyze a paper, since the time to perform each search is determined by load on the Internet, and Google specifically, rather than the capabilities of the user's own computer.

3.2 User Interface

SNITCH is implemented in Java, using Swing components for the user interface and the Google Search API for underlying search functionality. Figure 1 shows the main SNITCH user interface. The user creates a work list of documents to analyze, and is free to add and remove files from the work list, and to specify which of the listed documents are to be analyzed.

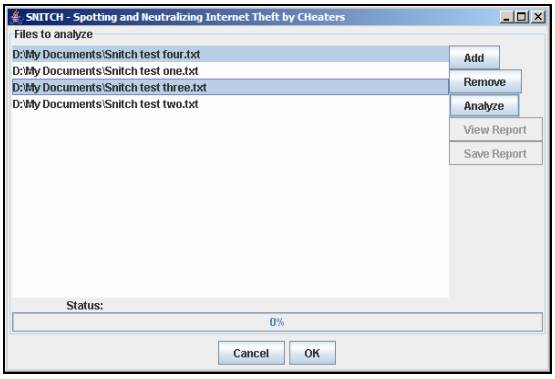

Figure 1. Main user interface of SNITCH software.

The currently version of SNITCH supports analysis of text documents only, so documents in other formats (e.g., Microsoft Word) must first be saved as text. A status bar provides visual feedback of progress during document analysis. Results of analysis can be viewed using the built-in HTML viewer or in any web browser (Figure 2).

3.3 Detection Engine

The detection engine analyzes the contents of the provided files looking for excerpts likely to have been plagiarized, as described in the above algorithm. The determination of which of these candidates will be searched for is based on the average word length in a sequence of W (or more) words, with searches performed for the top N candidates.

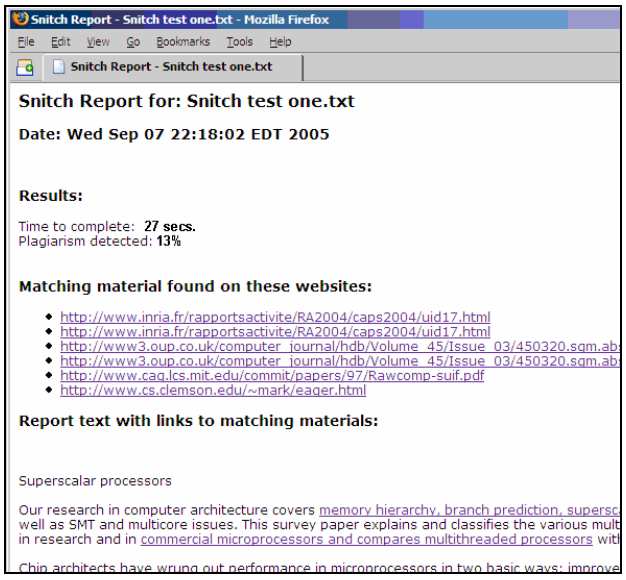

Figure 2. Example portion of SNITCH analysis report.

The Google Search API is the enabling technology for SNITCH. Google provides its API for free, subject to a straightforward license application procedure with Google Labs. There is generally a 1,000 search per day limit, which can be raised upon request to Google and reasonable justification. [3] As a practical matter, the number of searches performed (N) for each document analyzed by SNITCH is set low (N=20) by default to prevent exceeded the per day limit during periods of high activity.

Integration into a Java application is accomplished by downloading the Google API developer's kit, acquiring and installing a license key, and adjusting settings in the developer's programming environment of choice. Clear instructions are provided on the Google Web API web site. [3]

4. EVALUATION

An initial evaluation of the SNITCH software was performed to measure its effectiveness at detecting instances of plagiarism in custom-designed plagiarism benchmarks and a sampling of typical computer science student term papers. Results are compared with results for the same papers using the only other available practical and cost-effective software tool, Eve2. No formal comparison was done with online subscription services due to cost constraints.

4.1 Accuracy and Performance

Experiments using four synthetic benchmark term papers and a sampling of 10 actual student term papers were performed. The synthetic benchmarks consisted of carefully crafted documents containing known amounts and instances of cut and paste plagiarism representing hypothetical papers containing high, moderate, minimal, and no plagiarism. Actual student papers were manually analyzed using careful online detective work, and were divided into similar groupings based on the prevalence of plagiarism that was found. These student papers were all rough drafts, and any plagiarism detected was later removed by the students. Three experiments were performed to measure analysis speed and accuracy of SNITCH on the synthetic and real documents, and in comparison with the commercial plagiarism detection program Eve2 [2].

Table 1 compares the results of analysis of the synthetic benchmarks by SNITCH, using a window width (W) of 5 words and a candidate limit (N) of 20 searches. These values were selected because wider and narrower windows tended to lead to excessive false positives or a greatly reduced rate of detection, while trying to minimize analysis time. Known measurements are presented for comparison with the results of analysis of these benchmarks.

Percentage of plagiarism present or found in a document is based on a simple ratio of the number of plagiarized words to the overall word count for the document. Instances of plagiarism indicate the number of individual passages in the document that were plagiarized. Because these passages may exceed the window width W, more than one match may be found for a given instance. In these cases, only one match per instance was counted in tabulating results.

Table 1. Results of SNITCH benchmark document analysis.

	Manual stats		*Found by SNITCH*	
Benchmark	**Pct**	**Instances**	**Pct**	**Instances**
High	90	20	85	17
Moderate	50	12	83	10
Minimal	15	5	80	4
None	0	0	0	0

In all cases where cut and paste plagiarism was present, it was detected at least 80% of the time. Not surprisingly, benchmarks with more plagiarism present were more successfully analyzed than those with less present, suggesting that it is easier to catch the blatant cheater than the sly one. However, even minimal amounts of plagiarism were successfully detected.

Manual analysis of each student paper took approximately 30 minutes for a 7-10 page paper, which was found to be a point of diminishing returns for the manual identification of plagiarism in this set of papers. Detection of plagiarism in student papers was slightly less successful than in synthetic papers, but was still quite good. The range and average values for percent and count of instances of plagiarism known to exist in the papers and the results of SNITCH analysis are provided in Table 2.

Table 2. Results of student document analysis in SNITCH.

	Manual stats		*Found by SNITCH*	
Category	**Pct (avg)**	**Instances (avg)**	**Pct**	**Instances**
High	50-90 (75)	10-24 (19)	63	12
Moderate	20-49 (40)	4-13 (10)	50	5
Minimal	1-19 (15)	1-7 (5)	40	2
None	0 (0)	0 (0)	0	0

SNITCH was able to consistently detect 40-63% of the instances of plagiarism present in the papers, with slightly better success when more plagiarism was present. While detecting no false positives, SNITCH was able to positively classify all papers that contained plagiarism, and to provide a significant amount of concrete evidence of the cheating.

The commercial program Eve2 was used as part of the screening and grading process in a large number of student papers in a recent semester. While Eve2 was a significant improvement over manual analysis, Table 3 illustrates key differences as compared with SNITCH. Using the minimal amount of plagiarism detection for the same sample of 10 student papers, Eve2 detected less in significantly longer periods of time. In the case of papers with a high degree of plagiarism, SNITCH missed a small amount as compared with Eve2 (63% compared to 65% detection rate), although Eve2 required nearly 7 minutes of additional analysis. When an exhaustive search was used in Eve2, it was not uncommon for the analysis of a single paper to take 45-75 minutes without more than minimal improvement to the rate of detection. It is important to note that Eve2 produced many false positive results, incorrectly identifying passages as having been plagiarized, while SNITCH never produced false positives.

Table 3. Comparison of Eve2 and SNITCH.

	Avg. analysis time				*Avg.pct detection rate*			
Program	**High**	**Mod**	**Min**	**None**	**High**	**Mod**	**Min**	**None**
Eve2	7:30	7:00	6:45	6:45	65	27	12	1
SNITCH	0:44	0:38	0:18	0:15	63	50	40	0

The format of the results report produced by SNITCH provides basic statistics regarding detection, similar to those reported in Eve2 (search time, detection rate, etc.), as well as an HTML-ized version of the original document annotated with links to plagiarized sources, if any. Frequently, the links in the Eve2 report, which was in RTF format, were incorrect or did not correspond to the text to which the appeared to be linked. In developing SNITCH, every effort was made to make the report serve as a useful and accurate record of all plagiarism detected in an analyzed document.

5. CONCLUSIONS & FUTURE WORK

The problem of plagiarism of Internet sources is not going away anytime soon, and automated software tools are an effective means of detection. The SNITCH program provides an efficient and accurate alternative to commercial tools and services, producing acceptable accuracy and faster analysis at no cost; SNITCH will be made available for free download. Although development of SNITCH is ongoing, the current version provides a solid and usable tool to assist instructors in deterring and detecting cut and paste plagiarism.

Although a benefit specifically to computer science education is not yet proven, we believe that the availability of SNITCH will increase the threat of detection and continue to discourage students from resorting to cut and paste plagiarism. Because CS students tend to be technologically savvy, an unintended consequence of SNITCH may be its use as a tedious pre-screening, rewriting and detection avoidance tool. However, an unexpected benefit of this behavior is that the cheating student will spend more time with the material being studied. Obviously, our hope is that SNITCH is a deterrent rather than a catalyst.

Future planned extensions to SNITCH include support for analysis of MS Word documents, an improved search candidate identification algorithm, an adaptive approach to setting the window size and search limit based on document content, and analysis statistics that account for plagiarized passages that are larger than the search window. An investigation of more sophisticated semantic analysis algorithms for candidate identification is planned, although we are believers in the vaunted KISS principle. We hope to present a follow-up study in the near future, incorporating the experiences of a broad base of users and a more exhaustive set of test cases.

SNITCH is the result of collaborative student-faculty research at the Applied Computing Technology (ACT) Laboratory at Villanova University. The ACT Lab encourages diverse applications of computing technology to solve challenging problems, providing students with the opportunity to become invested in a significant research and development effort that builds on software engineering and capstone projects courses. [1]

6. ACKNOWLEDGMENTS

Thanks to Dr. Dan Joyce for his advice on approaches to manual search candidate selection which inspired the initial SNITCH algorithm, and to Dr. Way's recent students for providing a corpus of test cases.

7. REFERENCES

[1] The Applied Computing Technology Laboratory, Computing Sciences Department, Villanova University. 2005. Website: http://actlab.csc.villanova.edu.

[2] EVE Plagiarism Detection System. 2005. Website: http://www.canexus.com/eve/.

[3] Google Web APIs. Website: http://www.google.com/apis/.

[4] S. Grier. A tool that detects plagiarism in Pascal programs. Proceedings of the 12th SIGCSE Symposium on Computer Science Education, 13:1, pp. 15-20, 1981.

[5] C. Humes, J. Stiffler and M. Malsed. Examining Anti-Plagiarism Software: Choosing the Right Tool. Claremont-McKenna College technical report. 2003. Website: http://www.educause.edu/ir/library/pdf/EDU03168.pdf.

[6] Brian Martin. Plagiarism: policy against cheating or policy for learning? Nexus: Newsletter of the Australian Sociological Association, 16:2, pp. 1-12, 2004.

[7] D. McCabe. Levels of Cheating and Plagiarism Remain High. Center for Academic Integrity, Duke University. 2005. Website: http://www.academicintegrity.org.

[8] MyDropBox Internet Plagiarism Detection Service. Sciworth, Inc. 2005. Website: http://www.mydropbox.com.

[9] L. Renard. Cut and paste 101: Plagiarism and the Net. Educational Leadership, 57:4, pp. 38-42, 2000.

[10] R. Saikkonen, L. Malmi and A. Korhonen. Fully Automatic Assessment of Programming Exercises. Proceedings of ITiCSE'01, pp. 133-136, 2001

[11] R. Satterwite and M. Gerein. Downloading detectives: searching for on-line plagiarism. Colorado College, 2002. Website: http://www.coloradocollege.edu/Library/Course/downloading_detectives_paper.htm.

[12] TurnItIn Internet Plagiarism Detection Service, 2005. Website: http://www.plagiarism.org/.

[13] WCopyFind software. The Plagiarism Resource Site, University of Virginia, 2005. Website: http://plagiarism.phys.virginia.edu/.