An Automated System for Plagiarism Detection Using the Internet

Allan Knight, Kevin Almeroth
Dept of Computer Science
University of California
Santa Barbara, CA 93106-5110
{aknight, almeroth}@cs.ucsb.edu

Bruce Bimber
Dept of Political Science
University of California
Santa Barbara, CA 93106-5110
bimber@polsci.ucsb.edu

Abstract: In this paper we present a framework for plagiarism detection. Rather than exhaustively or randomly searching sentences in a student paper on the Internet for possible sources of borrowed ideas in student's text, we take a middle ground. With intelligent selection of sentences from papers we will show that examples of plagiarism can be found just as frequently and with fewer queries to Internet search engines. One of our goals is to use this technology to develop freeware that any instructor or teaching assistant, even modestly non-technical ones, can use to detect plagiarism in their classes.

Introduction

There is a simple fact that is sending a chill through academia: plagiarism in student work is on the rise. The Internet is only going to make the problem worse. Plagiarism is defined as presenting, "as new and original an idea or product derived from an existing source" (Merriam-Webster Online, 2003). Furthermore Plagiarism.org (Plagiarism.org, 2003) says that there are varying degrees of plagiarism. The range runs from copying someone's work verbatim to copy someone's ideas without proper citation. Students often do not, or choose not to, understand the most basic rule: if you borrow from someone's work, you must to give him or her credit. With the Internet making more text available for students to get ideas from, including actual services that offer plagiarized papers to students, detecting and preventing plagiarism is becoming an even greater challenge.

According to McCabe (McCabe, 2002), 10% of college students have copied several sentences from Internet sources without a proper citation. Also, 5% have turned in entire works from websites or so-called term-paper mills. The numbers are even worse among high school students with 52% having copied from Internet sources without proper citation, and 16% turning in papers fully from websites or term-paper mills. Worse yet, more students cheat every year (McCabe, 2002). Not only is it the problem of cheating in current classes but also the looming problem created by the tidal wave of students that believe plagiarism is a way to succeed.

To address this issue it is important for schools to actively deter, prevent and detect plagiarism in student papers. While honor codes can deter plagiarism (McCabe, et. al, 2002), they are not enough. To further deter students from cheating there must be a way for instructors to detect plagiarism. The traditional methods for detecting such cases are looking for changes in style or identifying passages that sound familiar. In order to then prove the suspicion, instructors must use a tremendous amount of intuition to exhaustively hunt for the reference. Newer methods are only slightly better. Instructors read the material, identify key phrases and search the Internet for these phrases. However, we believe an automated process will catch more students who attempt to cheat and make the process much faster. It is the intent of this paper to present a framework that will automatically catch those who choose to cheat.

We have identified four different approaches to plagiarism detection. These approaches differ in the number of queries that are required and in how effective they are at finding cheaters. By categorizing these approaches and evaluating their effectiveness, we hope to find an approach that combines the least number of queries with the best possible effectiveness.

Of the techniques we have considered, the most thorough is exhaustive searching. Searches done exhaustively mean that the entire text of a document is compared against the entire Internet. At one extreme a "window-based" approach can be used. With the window approach a window size is chosen and then is used to iterate through an entire document one word at a time searching the Internet for the words within each window. The number of queries required for this approach is on the order of the number of words in the document. A much more reasonable approach is to break the document into its individual sentences and querying either the entire sentence or fragments of the sentences. This reduces the number of queries to approximately the number of sentences. While these approaches are very effective at catching plagiarism, they are inefficient. Reducing the number of searches by being more intelligent about the input to the search engine is a much better approach and may be just as effective.

The first approach taken to reduce the number of queries is using surface linguistic techniques, specifically readability scores. By looking at the readability score of each sentence, the search can be narrowed to only those sentences outside a certain tolerance. Many instructors look for changes in writing styles to identify possible excerpts of student assignments that may be plagiarized. A change in readability scores may be indicative of style changes. While the surface features may do a good job of identifying suspect text, there may be other linguistic features that are even better at identifying possibly plagiarized excerpts. Features such as punctuation density or part-of-speech density may be even better indicators. By looking at these linguistic features we hope to be able to identify changes in writing style within a document.

Other deeper linguistic features that look at the syntax and semantics of a sentence may be even better. By looking at a student’s use of grammar, specifically incorrect usage, it may be possible to narrow the search to only those sentences that suddenly use the correct grammar. This has been used effectively for students who are non-native English speakers. From here we may progress even deeper into the linguistic features by compiling fingerprints of writing styles for each author. By looking at sentences that do not fit into the fingerprint, suspect text can be identified.

While all the previous approaches require queries on the order of a number of sentences or words, it may be possible to lower the number of queries some constant number per document, or better yet, per assignment. If the assignment subject is narrowed enough it may be possible to do only a few queries based on keywords used for the assignment. Once keywords are identified and searched for, the results can be downloaded and each assignment searched against this set of documents. The theory being that if a student is going to plagiarize from the Internet, it is most likely going to be from Internet sites that use the relevant keywords and that they will only look within the first 100 results. The challenge then becomes properly identifying the correct set of keywords. Analyzing each document and identifying the keywords associated with the entire document can either be done manually by the instructor or teaching assistant, or automatically.

Finally, at the other extreme from exhaustive searching, is random selection of sentences within an assignment. We believe that this is not a particularly effective approach, but unlike the other approaches discussed, requires almost no additional work. The idea simply involves randomly selecting a number of sentences from the assignment and then searching for those sentences, or fragments of them, in the Internet.

Having discussed a full spectrum of approaches, in this paper, we begin to analyze these approaches and better understand their effectiveness. In what follows, we present our initial framework for automatically detecting plagiarism and evaluate its effectiveness. First, we look at exhaustive approaches and then present others that select sentences to be inspected without exhaustively searching for every sentence in the Internet. The results of these processes are then presented and conclusions are drawn about their overall effectiveness. We also discuss areas of future research.

Related Work

The work most related to ours was originally started as a project by two Biophysicists at the University of California, Berkeley that eventually turned into a private venture: Turnitin.com, formerly plagiarism.org (Turnitin.com, 2003). While they seek to accomplish the same things that we are, they approach it in a much different way. Turnitin.com has its own servers with terabytes of information, collected from crawling the Web, subscribing to term paper mill sites, and collecting past papers from those who submit papers to be checked for plagiarism. Entire papers are then compared against this database of information.

Where we differ from them is in the philosophy of how to do the searches. While we both seek to find sources on the Internet, we feel their process is overkill. They can catch every single copied sentence but at a great cost of time as they search each sentence against all of the data they have collected. We instead look to intelligently selected sentences that are likely candidates and flag a paper as suspicious if those sentences are found in sources on the Internet. And then, like them, let the instructor or grader determine if the student has acted improperly in borrowing from a source. Our framework, like that of Turnitin.com, can easily be connected into any database of text, whether it is the Internet or a collection of past assignments.

One issue of concern for Turnitin.com is the legality of transferring a copy of each student's work to an outside organization. While some argue it violates the students' copyrights (Paulson, 2002), Turnitin.com reassures institutions this is not an issue since it falls into the category of fair use. However, we still believe that a) this issue has not been properly resolved, and b) it may also violate students' privacy. By employing an in-house system instead, it is more likely that these issues will not be of concern.

Another piece of related work is CopyFind (Plagiarism Resource Site, 2003). While its goal is also to detect plagiarism, its search domain is more limited. Rather than finding borrowed text from the Internet, it compares a collection of student papers to each other. If enough similarity is found between two papers, the papers are flagged for further inspection. CopyFind has no ability to determine if sources from the Internet were used.

Other related work deals with collecting digital media. SCAM (Shivykumar, 1995) and CHECK (Si, et. al, 1997) deal with finding similarities among documents in a common database. While their main focus is on finding similar documents in a file system or other databases of digital media, they are different from our work in one major respect. They look at similarities of documents as a whole, not at individual sentences. Usually if the documents are no more similar than 25% the same documents are not flagged. We are working in a context where similarity is not necessarily important, but identical text of maybe only a few sentences would be considered important. If we find that the average student paper is 100 sentences long and we want to flag all papers with at least five sentences that are possibly copied, then we are only talking about an overlap of 5%. Furthermore, these systems tend to not take any contextual similarities into consideration. This makes them easy to defeat by merely changing key words throughout the new document.

Still another body of work concentrates on finding plagiarism within programming assignments. For example MOSS (Boyer and Hall, 1999) and Sherlock (Joy and Luck, 1999) are two examples. However these solutions are similar to CopyFind in that they only look for duplicates among a single set of assignments and do not have the ability to search for versions outside of what is provided as input. Another well know system for detecting plagiarism in programming assignments is SID (Chen 2002). While it aspires to the same goal as the first two tools, it is quite different. SID uses a measure based on Kolmogorov complexity, which is universal. The truly unique feature of SID is the fact that because the Kolmogorov complexity is universal it is not, in theory, cheatable. While this system is targeted toward finding similarities in code, it could be altered to detect similarities in plain text as well. This system could then be very effective at detecting plain text plagiarism. However, this system, as it is now, is not meant to find similarities in text, which is the focus of this paper.

Automatic Plagiarism Detection

The ideas we are presenting here in using Internet search engines such as Google (Google, 2003) to search for text in students' work are relatively straightforward. There exists a wide spectrum of ways to determine a) whether any part of a student's work has been plagiarized, and b) which parts of the work are copied. The more interesting and more challenging question then is what parts of the work should be used in a search on Google. The choices range from using every window of *n* words in a document, to randomly selecting a part of a single sentence. While it would be expected that the exhaustive search would catch the most cheaters, we believe that a more selective process can be used to significantly reduce the number of searches that are needed and still catch the same, or almost the same number of cheaters. Below are different techniques we have developed and evaluated.

Exhaustive Searching

As mentioned before, exhaustive searching is probably the best and most effective way of detecting plagiarized work. However, there can be different definitions of what "exhaustive" means. One example would be to search for every combination of words in the paper. This though, would result in so many queries to Google, that it would take a *very* long time to process a single paper, not to mention an entire class full of papers. Furthermore, Google has strict rules about the number of searches than can be submitted per IP address per day. Instead, we use a simplified exhaustive search. This technique breaks every paper into each of its sentences and submits the first eight words from each sentence that is at least eight words long. This limits the number of queries to about 100 per paper. The number eight was chosen because it returns fewest false positives and avoids the problem of many documents containing common phrases in all text.

Once all the sentences are collected from each of the papers, they are sent as queries to Google and the results are saved. These results are then inspected for any possible instances of plagiarism. If any combination of one or more sentences returns a hit for some URL, it is flagged for further investigation. Here is where human investigation is critical. While this framework is automatic, it cannot make the final judgment as to whether or not plagiarism has actually occurred. It is up to the graders and/or the instructor to make the final judgment.

Narrowing the Search

We used the exhaustive search for a set of approximately 480 papers from a lower division college course in Political Science. Each paper was approximately 10 pages long. Using a true exhaustive search would have resulted in about 1,400,000 searches. Using our simplified exhaustive method, we narrowed the number of searches to about 55,000. The next step then is to look at more intelligent techniques to try and reduce the number of search queries even further. Given that there are quite a number of different possibilities, the challenge is to choose the one that is effective but does not require a huge number of searches.

Using Surface Linguistic Features

Linguists have developed may surface linguistic features over the years (Fry), (Gunning, 1952), (McLaughlin, 1969). These usually involve calculations based on the number of words per sentence and distribution of syllables in each word. While these linguistic features give no weight to the context of the sentence, research has shown they are effective in determining the readability of a given work.

As mentioned before one technique for manually detecting plagiarism is to look for changes in writing style. It is our hypothesis that readability scores can spot changes in style. By determining the readability score for each sentence and then choosing the scores below a certain level, it is hoped that the system will flag just as many papers for further inspection as the exhaustive search does. While intuitively it seems that the sentences with the higher grade level scores are the most likely candidates, this is in fact not the case. The grade level scores give some indication as to the readability of the text. Text that has been published in some form or another, tends to be more readable than that created by students. Therefore, we now believe that the most likely text to be plagiarized is not at the high end of the grade level score, but at the low end. This text will differ from what students traditionally write, which is at the high end.

For our process we chose the Flesch-Kincaid Grade Level (see Figure 1 for the exact formula) as our surface linguistic feature. Each sentence is processed by the UNIX command *style* (Haardt, 2002), which includes many different surface linguistic features. The score is recorded for each sentence and the sentences are then ranked after each paper is processed. All sentences with a grade level below 10 are then chosen to be sent to Google to see if they exist somewhere on the Internet. Just as in the simplified exhaustive case, all the results are saved and later analyzed to see if in fact the sentence was plagiarized.

$$(.39 \times ASL) + (11.8 \times ASW) - 15.59$$

where:

ASL = average sentence length (the number of words divided by the number of sentences)
ASW = average number of syllables per word (the number of syllables divided by the number of words)

Figure 1. Flesch-Kincaid Grade Level Score (Microsoft, 2003).

To summarize, our analysis strategy consists first of sending the keywords to Google. Second, the hits returned are downloaded from the Internet. Next, each paper is exhaustively searched against the downloaded pages. All hits are then recorded and any document that has hits is flagged and removed from the pool of papers that will be used to build further queries for the surface linguistic feature search. Because we are doing an exhaustive search locally, we can search the downloaded documents more thoroughly and have greater confidence in the findings. In our experience, this reduces the number of search by approximately 20 queries per paper flagged. Once the local search is done, the list of sentences is combined using the surface linguistic features. As before, we take all the sentences with Flesch-Kincaid scores less than 10, truncate it to include only the first eight words, and search Google for any hits. By combining the two processes it is hoped that more hits can be found with an even lower number of searches.

Evaluation

To evaluate the effectiveness of our proposed method we collected 480 papers from a freshman level political science course. These were turned in electronically in the form of Microsoft Word documents. Once we had the files we used *wvware* (Lachowicz, 2003) to extract the plain text from the files so that they could be processed more easily. After the text was extracted, the papers were broken into sentences for evaluation.

The programming language PERL was used to extract the sentences. Each line of text was read and after an entire sentence is extracted the first eight contiguous words are saved to an output file along with the name of the file from which it came. This output file was later used to compose the queries sent to Google. The sentences themselves were extracted using a simple regular expression that works for most sentence structures. Any quotations within a sentence are treated as part of the sentence, even if the quotation itself contains multiple sentences. When identifying the eight words to be searched, we only looked for eight contiguous words that were not within a quote. Therefore, while usually the first eight words are selected, it may be the case that eight contiguous words are not found until after a quote in a sentence. Searches for quotations are not done since it is assumed that they are properly cited. Therefore, verifying proper citations is outside the scope of this project, and it is left to the instructor.

Once all of the sentences are extracted from the papers, it was possible to search Google for possible plagiarism sources. Here again the PERL programming language is used along with the module libwww-perl (Aas, 2003). The previously generated output files become the input to this second program. The sentences are read, and sent to the Google web site within quotation marks. By putting the sentence in quotation marks we indicate to Google that we only want matches that have exactly the text for which we are searching. Also, the number of results returned is limited to 10. From our experience we noticed that most instances of actual plagiarism only return one or maybe two results. Those with more results are usually the result of searching for common sentence fragments that appear in other documents but are not plagiarized. As before these results are saved to a file for further processing.

The last step in the process is to analyze the output files from the previous step. Here we search for sentences that produce hits on the same web site as other sentences. Originally we thought to only flag consecutive sentences with hits on the same web site, however, we realized this would not catch all forms of plagiarism. And in fact the first paper we flagged as having sentences copied from an Internet source without proper citation would not have been caught using our original approach. In this case, the student copied six sentences from two sources and none of the sentences were consecutive. And while this approach does not catch those who choose to steal only one sentence from each source, this can be dealt with by extending the number of words in the search string. For now we are

only concentrating on those who copy multiple sentences from a single source.
We now have a comprehensive report on each paper and the possible hits that were produced from each. These reports can be inspected by the grader or instructor to determine if any student has improperly left out citations or has plagiarized the work of another.

Results

The results (see Table 1) justify further research into better ways in which to reduce the number of queries. Using the exhaustive searching technique, which involved over 55,000 queries sent to Google, we identified four papers that to one degree or another borrowed from other work without proper citation. One paper blatantly copied from another source. In it there was two entire paragraphs and three more sentences from three different sources. Another paper borrowed six sentences from two sources and spread those sentences throughout the paper. The last two examples involve the students copying two sentences from one document. In all cases there was a bibliography and other papers were cited, but none of the copied text was from papers actually cited. Therefore, it seems unlikely that the students did not know how to do proper citations or simply forgot.

Type of Search	**Number of Queries**	**Number of Possible Hits**
Exhaustive Search	55,522	4
Narrowed Search	18,152	3

Table 1. Results collected from testing Framework

When we reduced the number of queries to 18,000 using the surface linguistic features, we got similar results. However, not all of the plagiarized sentences were found. This was because some of the copied sentences had Flesch-Kincaid grade levels higher than ten. The end result was that all but one of the papers flagged in the exhaustive search were flagged by the Flesch-Kincaid system as well. However, because the paper that was not caught only had two sentences in it that were copied, we are quite pleased with this result. Realistically, we are uncertain whether a paper with only two plagiarized sentences would actually be prosecuted as a cheating offense. Therefore, the fact that we would not have identified this paper as possibly being plagiarized is not particularly significant.

Conclusion and Future Work

While we would like to see a greater reduction in the number of queries in future work, we have shown that the number of queries can be reduced. By choosing sentences intelligently rather than blindly searching each sentence exhaustively, we have shown that it is possible to catch almost all of those who chose to cheat. In the future we would like to see an even greater reduction in the number of queries and at the same time catch all of the cheaters.

The next step is to find ways to better select possible sentences for submission to Google. While the surface linguistic features look promising, metrics that look deeper into the context of each sentence are needed. These contextual metrics could be anything from part-of-speech densities, such as the number of pronouns or number of adjectives, to the density of stop words such 'the' or 'an'. We have shown that we can reduce the number of queries, now we must refine the process to reduce them even further and retain our accuracy in identifying improperly borrowed text.

Because we do not know which contextual metrics are the most revealing, it may be better to use pattern recognition techniques. Well-known pattern recognition techniques are based on this principal. By counting the densities for all interesting metrics, those that are not good at differentiating authors will become less significant while those that are better at differentiation will become more significant. Exploiting pattern recognition features will perhaps lead to a better technique to identify potentially plagiarized sentences in the future and further reduce the number of queries required. While this process may not catch all cheaters, it will catch some and further reduce the number of queries that the previously mentioned process needed. Once a paper is flagged as being possibly plagiarized, there is no need to send any queries to Google for that paper. On average we would expect to save 10 to 20 queries per paper flagged with this process and in some cases no queries at all.

Acknowledgements

The authors wish to acknowledge the contributions of the following people Stan Nicholson, George Michaels, Alan Moses, Joe Navarro, Brandon Brod, and the numerous other faculty at UCSB who has offered comments and/or agreed to participate in future efforts. Without their contributions and ideas this paper would not have been possible.

References

Aas, G. (2003). lib-www-perl-5.64, http://lwp.linpro.no/lwp/.

Bowyer, K.W., & Hall, L.O. (1999). Experience using "MOSS" to detect cheating on programming assignments, *Frontiers in Education Conference*, 1999. FIE '99. 29th Annual , Volume: 3 , 10-13 Nov. 1999 Page(s): 13B3/18 -13B3/22 vol.3.

Chen, X., Li, M., Mckinnon, B. & Seker, A. (2002). A Theory of Uncheatable Program Plagiarism Detection and Its Practical Implementation, http://genome.math.uwaterloo.ca/SID/.

Fry, E. Fry's readability graph: clarifications, validity and extension to level 17, *Journal of Reading*, vol. 21, School of Education, Open University.

Google (2003). http://www.google.com/.

Gunning, R. (1952) *The Technique of Clear Writing* , McGraw-Hill.

Haardy, M. (2002). diction 1.02, http://www.gnu.org/directory/diction.html, GNU.

Joy, M., & Luck, M. (1999). Plagiarism in programming assignments. *Education, IEEE Transactions on* , Volume: 42 Issue: 2 , May 1999 Page(s): 129 –133.

Lachowicz, D. (2003). wvware, wvText 1.0.0, http://sourceforge.net/projects/wvware, SorceForge.

McCabe, D. (2002). Cheating: Why Students Do It and How We Can Help Them Stop. *American Educator* (2001-2002) winter, 38-43.

McCabe, D., Trevino, Linda K., & Butterfield, Kenneth D. (2002). Honor Codes and Other Contextual Influences on Academic Integrity. *Research in Higher Education*, Vol. 43, No. 3, June 2002, pg. 357 – 378.

McLaughlin, H. (1969) SMOG grading - a new readability formula. *Journal of Reading*, 22, 639-646.

Merriam-Webster Online. (2003). http://www.m-w.com/cgi-bin/dictionary?book=Dictionary&va=plagiarism.

Microsoft. (2003). Readability Scores, *Spelling and Grammar*. http://office.microsoft.com/assistance/preview.aspx?AssetID=HP051863181033.

Paulson, L. (2002). Professors Use Technology to Fight Plagiarism. *IEEE Computer*. 35(8): 23-25.

Plagiarism.org. (2003). "Types of Plagiarism", http://www.plagiarism.org/research_site/e_what_is_plagiarism.html.

Plagiarism Resource Site (2003). *CopyFind*, 1.2, http://plagiarism.phys.virginia.edu/software.html.

Shivakumar, N., & Garcia-Molina. (1995). SCAM: A Copy Detection Mechanism for Digital Documents. *The 2nd International Conference on Theory and Practice of Digital Libraries.*

Si, A., Leong, Lau, H., & Rynson W. (1997). CHECK: A Document Plagiarism Detection System. *ACM Symposium for Applied Computing*, pp.70-77, Feb. 1997.

Turnitin.com. (2003). http://www.turnitin.com.