Sentence-Based Natural Language Plagiarism Detection

DANIEL R. WHITE and MIKE S. JOY
University of Warwick

With the increasing levels of access to higher education in the United Kingdom, larger class sizes make it unrealistic for tutors to be expected to identify instances of peer-to-peer plagiarism by eye and so automated solutions to the problem are required. This document details a novel algorithm for comparison of suspect documents at a sentence level and has been implemented as a component of plagiarism detection software for detecting similarities in both natural language documents and comments within program source-code. The algorithm is capable of detecting sophisticated obfuscation (such as paraphrasing, reordering, merging, and splitting sentences) as well as direct copying. The implemented algorithm has also been used to successfully detect plagiarism on real assignments at the university. The software has been evaluated by comparison with other plagiarism detection tools.

Categories and Subject Descriptors: K.3.1 [**Computers and Education**]: Computer Uses in Education—*Computer managed instruction*; E.5 [**Files**]: Sorting/Searching; I.5.4 [**Pattern Recognition**]: Applications—*Text processing*; J.5 [**Arts and Humanities**]: Linguistics; K.4.1 [**Computers and Society**]: Public Policy Issues—*Intellectual property rights, ethics*

General Terms: Algorithms, Languages, Human Factors

Additional Key Words and Phrases: Natural language, plagiarism detection

1. INTRODUCTION

Plagiarism has become an increasing problem in higher education institutions in recent years. The problem is especially prevalent in British institutions because of a governmental target of having 50% of the population study in some form of higher education [Curtis 2003]. This has led to rapid increases in class sizes at most British universities, which, in turn, has led to an increasing problem with detecting plagiarism among students. A further contributing factor is that students are now much more comfortable with Information Technology and the Internet, allowing them to find and copy sources with ease [Carroll and Appleton 2001].

This is a subject that has received much attention both in the press and in academic circles. Many useful tools have been developed to detect plagiarism in both programming and essay-based assignments. The main aim of these

Authors' address: Department of Computer Science, University of Warwick, Coventry, CV4 7AL, United Kingdom; email: {D.R.White, M.S.Joy}@warwick.ac.uk.

programs is to guide the investigator toward submissions that are most likely to have been plagiarized. Comparisons of such software tools are available from other sources, including Chester [2001] and Culwin et al. [2001], as are detailed studies of individual tools and their algorithms [Culwin and Lancaster 2001; Finkel et al. 2002; Hoad and Zobel 2003; Joy and Luck 1999; Monostori et al. 2001, 2002; Prechelt et al. 2002; Ribler and Abrams 2000]. Carrol and Appleton's *Good Practice Guide* [2001] provides guidelines for non-information technology approaches to reducing plagiarism. Decoo's [2002] discussion of the whole process of dealing with academic misconduct, from detection through to punishment, has more of a focus on postgraduate and research misconduct than is usual in the other sources.

In natural language, the sentence can be considered as one of the building blocks for the communication of ideas. With this concept as a starting point, we have designed an algorithm to compare documents at a sentence level, as plagiarism will be likely to occur by reworking a source text on a sentence-by-sentence basis. The algorithm described in this article is intended for a thorough pair-wise comparison of a set of documents, whether they comprise a particular class' essays, the comments from a set of programming assignments, or an article along with sources that it is likely to have plagiarized. The results generated highlight places where two documents are very similar, in a manner that makes it easy for a teacher/academic to determine whether an investigation of alleged plagiarism is worth pursuing. In addition to the detection algorithm, a filter was also designed to be applied to the results when viewed. This filter applies a heuristic approach to deciding whether similarities are useful based on the number of similarities to another sentence. The results and the visualization method employed make the suspicious texts easy to compare for the teacher, although the implementation of the proposed algorithm is not especially fast when compared to a more mature system such as *CopyCatch* [Woolls 2004].

The proposed algorithm has been integrated with *Sherlock*, a source-code plagiarism detector developed at the University of Warwick [Joy and Luck 1999], that has been used to successfully detect programming plagiarisms in the past. The new version of *Sherlock* has the capability to compare documents in plain text form as well as having a more sophisticated approach to detecting plagiarism in the natural language (commented) sections of programming assignments. A thorough testing of the implementation's capabilities has been conducted in order to determine the effects of the various parameters to the algorithm on the results produced. These new features of the program have been tested on programming and essay assignments set on courses in the Department of Computer Science at Warwick in the last year and successful identification of possible plagiarisms has occurred as a result.

A direct comparison of Sherlock with two other freely available natural language plagiarism detectors was undertaken. This comparison shows that while Sherlock is lacking in speed during the detection, it compares favorably in both quality of results and the method of results visualization.

2. A SENTENCE-BASED DETECTION ALGORITHM

When detecting collusion between peers, there are two main classes of algorithm currently in use. The "fingerprint" approach [Finkel et al. 2002; Hoad and Zobel 2003; Monostori et al. 2002] involves creating a small sample of a document that can be indexed and then be used to search rapidly through other documents that are likely to contain high levels of similarity. The other approach is a thorough pairwise comparison of a group of documents, which tends to produce more readable results at the cost of speed in the comparison stage [Culwin and Lancaster 2001; Monostori et al. 2001; Ribler and Abrams 2000].

A decision was taken to attempt an algorithm of the second class, as this was more in keeping with the approach already used by Sherlock for source-code plagiarism detection [Joy and Luck 1999]. Furthermore, a thorough comparison algorithm is a prerequisite for a fingerprinting algorithm, as the better results generated by the former are useful in comparing suspicious documents found by fingerprinting [Finkel et al. 2002]. The algorithm was, therefore, designed to meet the following objectives:

1. Results must be output in a form that can be *easily visualized* for interpretation by a human marker.
2. Results must *limit the amount of false information*, where false information is defined as similarities that are not useful in determining if plagiarism has occurred.
3. The comparison must be *thorough*, in order to reduce the chance of missing an instance of plagiarism.

A suspicious document can be broken up into its constituent sentences. Sentences can be thought of as the components of which a natural language is comprised, similar to statements in a programming language. Taking a higher level view, such as paragraphs, was judged too coarse as the levels of similarity between paragraphs would be much higher than between individual sentences. Lower-level views, such as words or *n-grams* of characters [Ribler and Abrams 2000], were judged too fine a measure as they do not have the same scope for identifying rewording of a source.

2.1 Preprocessing Documents

The preprocessing stage of the algorithm is to read in the documents that are being compared and parse them into *Document* objects that contain a list of *Sentence* objects, which, in turn, each contain a list of *words* that were found in the original sentence in the source text.

The list of words contained in a Sentence object is then subject to three filters. First, all words are converted to lower case to save time in later comparisons. Second, a list of words that are considered too common to be useful are never stored in the processed form of the document. This list is passed to the parser as a parameter of the detection engine and will generally consist of words that add no meaning to the sentence, such as "the," "a," or "that." It is noted that

—***Sentence 1: "The brown dog was feeling very tired"***
Sentence 2: "While the grey cat was full of energy, the brown dog was feeling very tired"
—***(Common and repeated words removed)***
SentenceObject 1: {***brown, dog, was, feeling, very, tired***}
SentenceObject 2: {***while, grey,cat, was, full, energy, brown, dog, feeling, very, tired***}
—***The objects have 6 words in common, which is 100% of Sentence 1 but only 55% of Sentence 2.***
Average similarity is 77.5%.

Fig. 1. Similarity score example.

Hoad and Zobel [2003] refer to these words as *closed-class* words and suggest that they are of little use in similarity searches. The third filter removes words that are repeated in a sentence so that they only appear once in that sentence's list. This is done because of the nature of the metric used when comparing sentences, which is explained below.

2.2 Comparing Documents

At this stage, alongside the original source-texts there is a set of objects containing a specialized, processed form of the originals. These documents are compared pairwise by comparing every sentence in the one document to every sentence in the other. The comparison is done by computing a *similarity score*, based on the word count metric detailed in Culwin and Lancaster [2001]. The score is the average similarity between the sentences, computed as a function of words in common and the lengths of the sentences being compared. The example in Figure 1 shows how this works in practice.

A parameter, the *Similarity Threshold*, is passed to the detection engine. Any similarity scores greater than or equal to the Similarity Threshold are then stored as a link between the two sentence objects. The score used here is not sufficient in itself to catch cases where sentences have been merged into a longer sentence or split into shorter sentences during the act of plagiarism, as the differing lengths affect the average similarity. For this reason, an extra parameter is used to "catch" such cases. The *Common Threshold* is an arbitrary number that says that, regardless of the score, any two sentences with a number of words greater than or equal to the Common Threshold in common will cause a similarity to be stored. By default (due to testing results), the Similarity Threshold is set to 80 while the Common Threshold is set to 8, although experience suggests that setting the Common Threshold as low as 6 is useful when comparing source-code comments or small groups of documents as the increased amount of similarities does not detract from the quality of the results in these situations. This comparison process is summarized in Algorithm 1.

2.3 Filtering Results

After the comparison process has completed, the result is a data structure containing many links between sentences in different documents. Because of the nature of the task being carried out, it is likely that not all of these links will be useful. Consider the case where a tutor is comparing essays submitted on the

Algorithm 1 . Pseudocode for the Comparison Algorithm

```
Document[] docs = readDocsFromDisk();
for each Document, i, in docs {
  for each document,j, following i in docs{
    compareSentences(docs[i], docs[j]);
  }
}
compareSentences(Document doc1, Document doc2){
  for each sentence,i, in doc1{
    for each sentence,j, in doc2 {
      int common = number of shared words;
      int score = similarityScore(i,j,common); // see example
      if(score > SIM_THRESHOLD ||
        common > COM_THRESHOLD)
        storeLink(sent1, sent2, score);
      }
    }
  }
}
```

same title. Many of the similarities between these documents will be caused by students using the same sources, the majority of which would be correctly quoted. In such a situation, the task of identifying the most likely plagiarized documents can be simplified by filtering out the similarities that are most common, such as those references. The chosen method is to set a threshold for the filter, such that any sentence with more than the threshold number of similarities will then have all its similarities ignored. By default this value is 6.

2.4 Document Scores

A score is assigned to each document so that pairs of documents can be compared to each other. Those with high scores are then the ones that it would be most useful to examine. There are two possible ways of calculating the document's overall score in the software. The first is to assign the score as the value of the similarities between it and its most similar document. The second is to multiply the total similarity score by the percentage of the total that came from its most similar document. The first alternative often causes the highest scoring documents to "pair up" (i.e., the documents tend to form into pairs with the same scores near the top of the list) and it is then easy to see where likely plagiarism pairs occur. The second just ensures that documents with a high instance of similarity to one document will have their score increased by more than those that do not. Either way, the assumption made is that collusion is more likely to have occurred between two students than between widespread groups. Anything matching that pattern is more likely to be a deliberate plagiarism than one in which the similarities are spread across the whole data set, which could be caused by common sources or bad paraphrasing and would not necessarily constitute plagiarism if the sources were cited correctly.

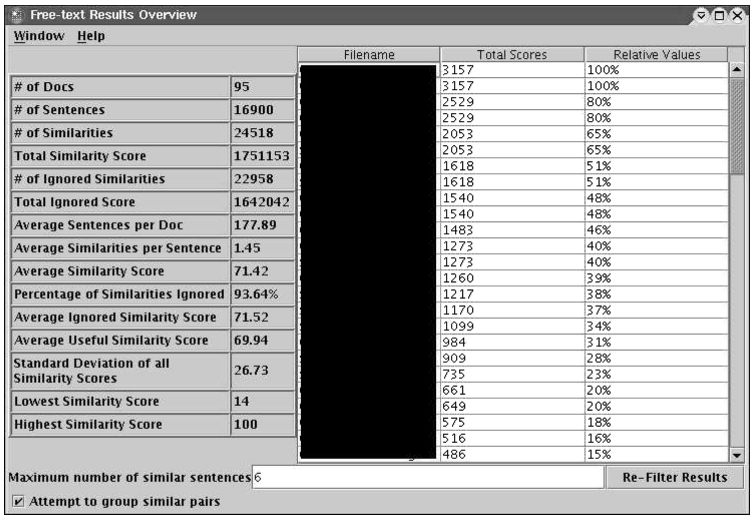

Fig. 2. Initial results screen.

2.5 Complexity of the Algorithm

The time for the algorithm to complete is $O(n^2)$, where n is the number of documents being compared. In the case where one document is being compared to a group of likely sources, it will reduce to $O(n)$. Optimizations are possible to reduce the constant factors of the time taken, such as using hash tables to store lists of words and loading all the documents into memory before commencing comparison, and work to further optimize the algorithm is ongoing.

3. VISUALIZATION OF RESULTS

An important feature of any plagiarism detection tool is the method of visualizing results. Culwin and Lancaster have devised an innovative image-based method to visualize their "fragmentary intercept similarity score matrix," for their "Visualisation and Analysis of Similarity Tool," (VAST)[2001]. Ribler and Abrams [2000] proposed their Categorical Patterngram as a way of visualizing a many-to-one comparison on a chart. The Culwin and Lancaster method allows a comparison of a single pair of submissions, while the Ribler method allows a many-to-one view but forfeits the detail shown by a one-on-one viewer. The method devised for viewing the results in Sherlock had the aim of providing both a many-to-one and a one-to-one results view, as well as behaving similarly to the hypertext approach used elsewhere [Monostori et al. 2001; Prechelt et al. 2002].

The initial screen presented to users lists all documents along with their scores and some statistics about the comparison results. From this screen the user can decide which documents need examining. It is very rare that a user will have time to examine all the possible pairs from here: in a set of 100 essays, for example, there are 4950 pairs to examine. For this reason the user will probably only examine the documents close to the top of the list.

Figure 2 shows the results from a set of student essays used during testing. The black box drawn over the screen is in accordance with data-protection

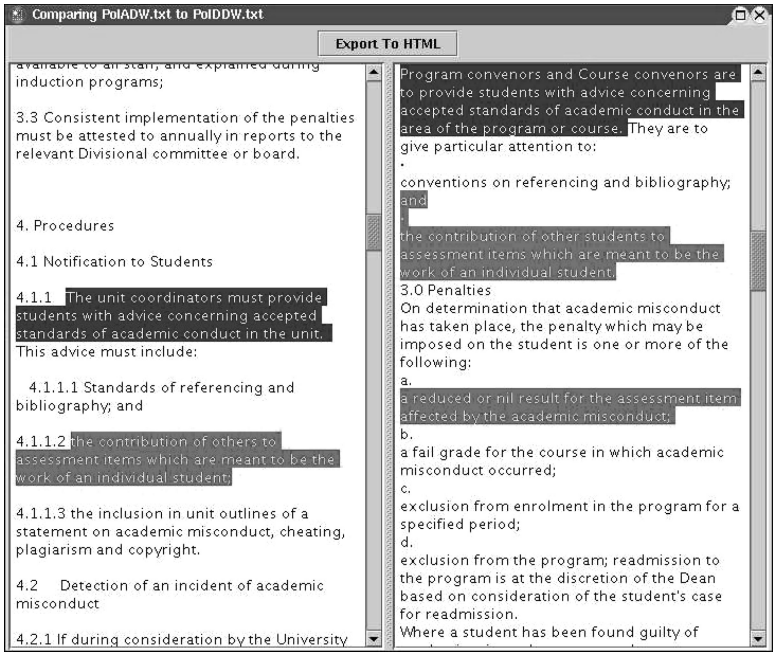

Fig. 3. One-on-one results visualization.

requirements. The screen shows the document's final scores as well as some statistics about the set as a whole. As can be seen from Figure 2, the vast majority of the similarities found by the detection algorithm are ignored in a large set—93.64% in this case. This is normal for a large data set and ensures that only those similarities most likely to be useful have been kept, as was intended in the specification of the system. The other pattern shown by the top of the list having much higher scores than the rest is also repeated in most student assignments, suggesting that intracorpal plagiarism is not as widespread as we might have feared from the press coverage, although the scores produced do not offer any indication of the level of plagiarism of external sources.

A many-to-one results viewer is made visible when the user selects a document from this table. The viewer displays the selected document on the left-hand side and highlights suspicious sentences in red. By clicking on these sentences, the user can choose the link they wish to follow from that sentence. The chosen document will be displayed on the right-hand side of the screen and the display is then scrolled to make the similar sentence visible. In this manner, the user can view all the documents that have similarities to the one document they are examining.

A one-on-one viewer is also provided that only shows the similarities between the chosen pair. This is most useful in deciding if collusion has occurred between two students. It also allows the user to output HTML frames that can be used on computers that do not have Sherlock available or printed to form evidence, for example, in a formal hearing to investigate alleged plagiarism.

4. TESTING THE PARAMETERS

As has been described previously, there are various parameters to the detection engine that can be set by the user. These include the list of common words, the Similarity Threshold, the Common Threshold, and the maximum links setting for the filter. It was desirable to carry out testing in order to determine some set of "optimum" values for these parameters in the majority of cases. In carrying out the tests an opportunity was also available to assess the effects of the variations in settings in order to see which parameter was most "dominant."

In order to achieve this goal, a definition of what is considered the optimal result is required. A *coderivative* is defined to be a pair of documents that can be shown to have a common source or sources and a similarity measure is defined to be a way of measuring the possibility that a pair of documents is coderivative [Hoad and Zobel 2003]. Hoad and Zobel [2003] state that any similarity measure is ideal if, "When searching for co-derivatives, no document should be ranked more highly than the one that is identical to the query," where the query is defined to be the document being compared to the source.

By the above definition, the similarity measure described so far is not ideal, since an exact copy of the original may not return a higher score than a document where the original has been copied and pasted in its entirety more than once. However, this would be a counterproductive measure for a plagiarist to take since it would not only increase the similarity between their document and the original but the repetition does not make sense in terms of good style. Therefore, for the purposes of these tests the definition of an ideal similarity measure [Witten et al. 1999] is redefined to be one that returns the documents most likely to be coderivative as having the highest scores. In the case where two similarity measures are both ideal, the better measure will be judged by the accuracy of the returned similarities.

Accuracy is not assigned a numerical value since whether a pair of sentences are plagiarized is, in large part, a matter of subjective opinion. It is not the purpose of the software to state categorically whether plagiarism has occurred but merely to provide an indication of which documents are most likely to be plagiarized. Accuracy will, therefore, be determined by comparing the results to the default settings used during development and declaring whether they show the suspected plagiarism more or less clearly.

4.1 Issues with the Analysis of the Natural Language Plagiarism Detection

There can be no definitive way of saying that all possible plagiarism has been detected within a large data set. In defense of the detection engine, there have been no noticeable incidents of Sherlock missing a document plagiarized from a student within the same set, although this cannot be taken as proof that it will never miss or has never missed an instance of plagiarism. Also, the term accuracy, as defined above, is a subjective judgment reliant on the tester's concept of what constitutes definite plagiarism. However, it is unlikely that a computer program would ever be completely trusted to automate the task of plagiarism detection, verification, and punishment, since a human judgment will always be needed for a serious charge.

In matters such as this, plagiarism detection is not amenable to definitive analysis and it is, therefore, necessary that human judgment will play a role in choosing which settings produce the *best* results.

4.2 The Tests

4.2.1 *Default Settings.* These are the settings originally used during development of sentence-based plagiarism detection for Sherlock. (Explanations of the purposes of each individual parameter are given earlier in this paper).

Similarity Threshold	80
Common Threshold	6
Common Words	the, an, and, a
Maximum Links	6

These seemed, by observation, to be the weakest set of parameters that returned accurate results. It was the purpose of the following tests to determine whether different parameters returned more accurate results and to, therefore, decide which parameters to set as the defaults. All subsequent tests will change one of the default settings and analyze how the results differ from those produced by these settings.

4.2.2 *Data Sets.* Tests were conducted using two sets of anonymized essays. The first had a set title while the second afforded students a choice between two titles. For the purposes of testing Sherlock, these assignments were ideal as having the whole class write on the same title makes the opportunity for collusion much greater.

From this point on, the data sets will be referred to as follows:

- The first set of essays, will be referred to as set A and consisted of 125 documents.
- The second set of essays had two possible titles. Those submitted on the first title will be referred to as set B1, which contained 95 documents. The other will be referred to as set B2 and contained 36 documents.

Students were asked to submit the work online [Joy and Luck 1998] along with a paper copy. The requirement to submit electronically was not enforced because it was the first time this had been required for a nonprogramming assignment and was therefore being used as a *dummy run*.

Set A contains a plagiarism triple, with a very strong similarity between one pair of the three. A couple of apparently false reports occurred in this set when one pair included the same document as an appendix while another had very similar bibliographies (which could also have been pursued as an indicator of possible plagiarism). Set B1 contained four very similar pairs, with one of the pairs having a third member partially similar to the other two. Again, there is also a good match between students with similar bibliographies. Set B2 has one definite pair and another small instance is also visible, where students may have copied each other or may have plagiarized the same source. These plagiarisms were all identified using the default settings. It was also noted

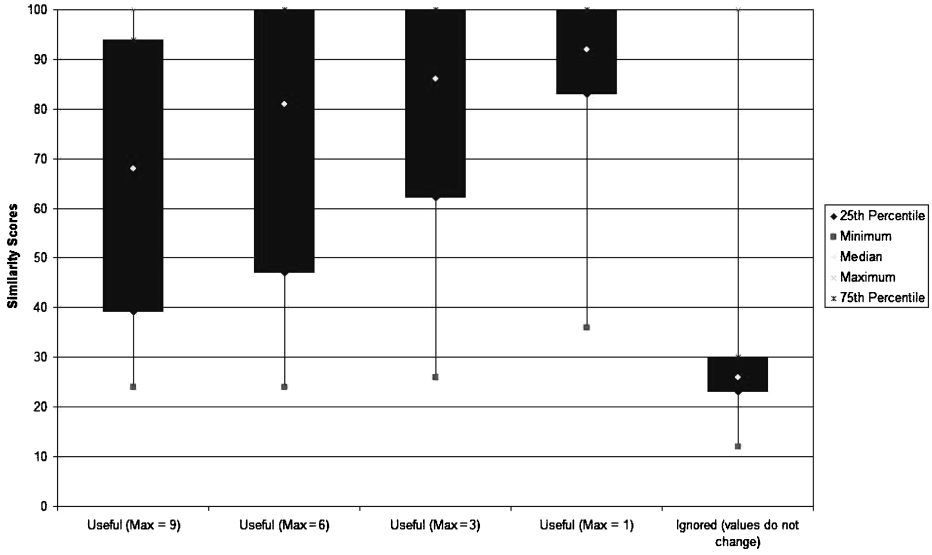

Fig. 4. How the filter affects similarity score distributions (default settings).

that the similarity sets generated for the two large data sets (A and B1) were unreasonably large. The theory to explain this was that the default settings were causing too many unhelpful similarities to be stored in such large sets.

For the most part, the tests were run on set B2 for the sake of speed, as a complete run could take place in a couple of minutes, while the larger sets were taking around 25 minutes per run. When a new setting appeared sufficiently beneficial it was applied to the other sets to ensure that the changes were not an artifact of the data in B2. In cases where individual files are referenced from the text, they have each been given a unique ID for the set to hide the identity of the students involved.

4.2.3 *Varying the Strictness of the Filter.* The default setting for the filter is the least important of the four settings, as it can easily be changed by the user after the detection engine has finished comparisons. It is usual in a medium to large data set for the vast majority of similarities to be ignored by the filter. This is normally acceptable because we are interested in the places where the similarities are limited to a few documents rather than the case where many have similarities to a given sentence because of lecture notes or recommended texts.

Figure 4 shows how the filter affected the useful score distributions for set B2. As would be expected, the filter has a large effect on the similarities that are considered useful. However, because of the large amount of similarities being ignored, the score distribution for the ignored scores stays roughly the same regardless of setting.

Figure 5 shows how the same settings affected the relative values of document scores. The top document had a value of 100%, in each case, with all others given a value as a proportion of their score compared to the top score. The file

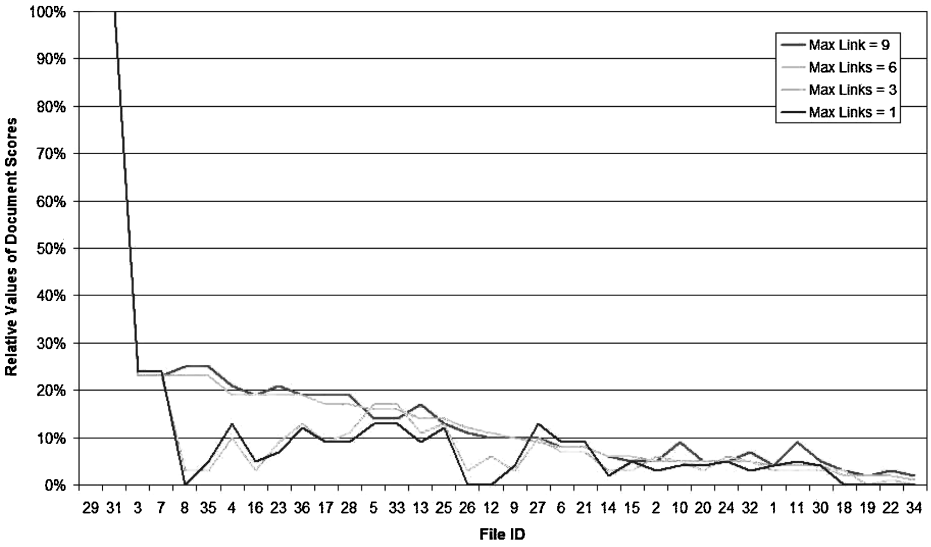

Fig. 5. How the filter affects relative values on default results for set B2.

IDs are placed in sorted order for their position using the default settings. The one definite instance of plagiarism is very well separated from the rest of the documents in terms of score, while the smaller instance of plagiarism in the pair of 3 and 7 also remains in the same position regardless of the setting. The documents after these two show no obvious signs of plagiarism so we can say that the accuracy of the filter improves as it becomes stricter. However, stricter filter settings also resulted in more similarities being ignored and so the results did not show all the obvious plagiarism.

For this reason, it was decided to keep the default filter setting as it was originally.

4.2.4 *Varying the Similarity Threshold.* This setting will store any sentences with a high similarity that do not have enough words in common to be stored by the common threshold check; this ensures that similar short sentences will be stored. Figure 6 shows how varying the Similarity Threshold affected the useful score distributions in data set B2. This setting has a far more drastic effect than the filter settings. On a very low Similarity Threshold, many more similarities are stored between sentences that would not appear similar to the eye and this results in a much less accurate set of similarities being considered useful by the filter. From the diagram, it would appear that the previous default of 80 is best as it produces a distribution where over 50% of the useful similarities have a score above 80, and at least 25% have a score of 100, indicating virtually identical sentences in terms of their content.

When the Similarity Threshold is set to 95, the results do not follow the pattern, one would expect as the distribution becomes more spread. This is because there are many sentences with scores between 80 and 95 that do not get stored by the Common Threshold, so setting the Similarity Threshold any

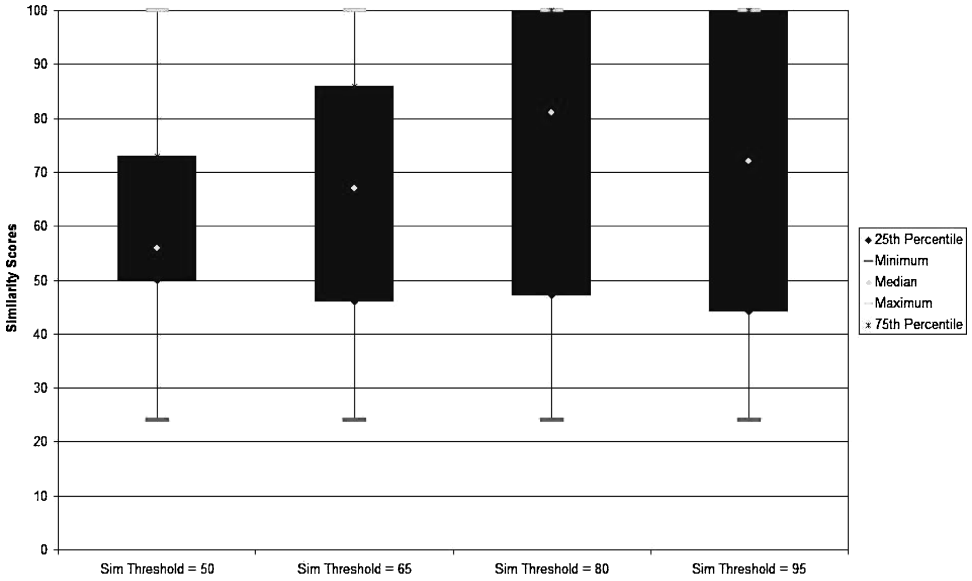

Fig. 6. How the Similarity Threshold affects useful score distributions.

higher is actually counterproductive as it results in fewer accurate similarities being passed to the filter.

Upon investigation of the results produced, the Similarity Threshold had very little effect on the order of the documents, but lower settings produced far less accurate results in that marked similarities were not very similar at all. For this reason, the Similarity Threshold was kept at its previous default value.

4.2.5 *Varying the Common Threshold.* The effects caused by varying the Common Threshold were much more drastic than those caused by varying the Similarity Threshold and this confirms the notion that the Common Threshold is the more dominant metric. Figure 7 shows this very well. Setting the Common Threshold lower than the original default served no real purpose at all: so many similarities were stored that only 117 out of 650,000 were considered useful by the filter! However, while the results themselves were of little use, it is interesting to note that the obviously plagiarized pair was still at the top of the list. Toward the stricter end of the settings, increasing numbers of scores are kept as a percentage of the total as well as increasing the overall useful score.

Looking at the results, the setting of 8 produced more accurate results than the lower settings. The results produced using this setting actually had another couple of pairs that had probably copied from the same source without a citation. Extra plagiarism was also found in set B1, but the difference was less noticeable in set A, which seems to have a lower amount of plagiarism.

From these findings it is clear that when using an increased Common Threshold the results produced became more accurate.

4.2.6 *Varying the Common Words List.* These are the words that are filtered out by the parser during preprocessing of the documents. These words

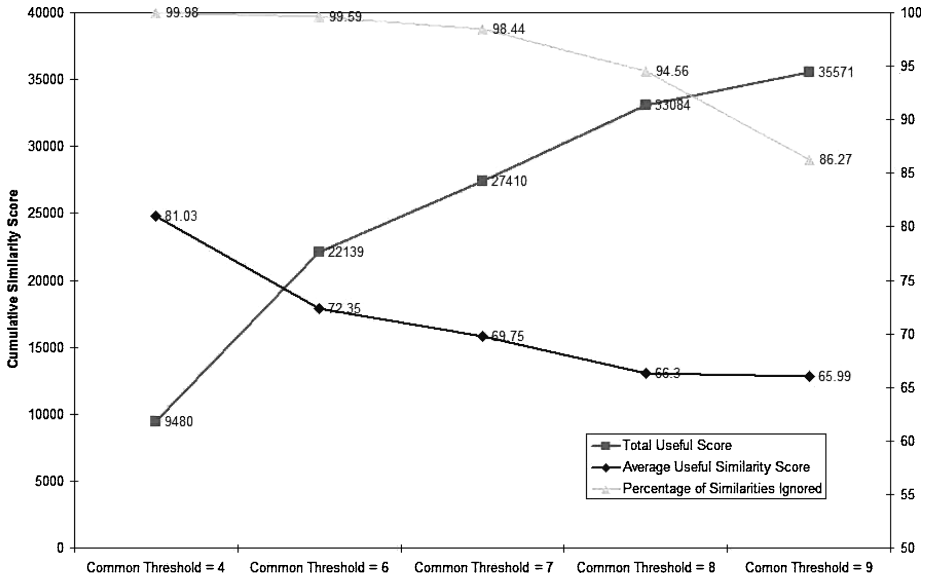

Fig. 7. The effects of varying the Common Threshold.

are judged too common to be useful in distinguishing between sentences. Originally, the intention had been that nouns common to an assignment could be added to this list, but upon trying this it was found to have too drastic an effect, resulting in far less accurate results. The different lists used to test the detection engine are shown below in the tabulation:

No common words	
Default list	the, an, and, a
Words with no meaning	the, an, and, a, so, as, of, or, to
Many common words	the, an, and, a, so, as, of, or, to, its, this, that, in, this, hers, these, we, they, do, so, be, if, for, any, on, is, was, out, are

The long list and the empty list were included as a check that the common words list was having a worthwhile effect on the results. By having no words filtered we would expect many more similarities while filtering too many would greatly reduce the accuracy of the results. The other non-default list was included because after the original hunch that excluding common words was corroborated by Hoad and Zobel's [2003] observation that "[closed-class words] indicate the structure of the sentence and the relationships between the concepts presented, but do not have any meaning of their own." This suggested that filtering more of the closed-class words may result in more accurate results.

Figure 8 shows a very similar shape to the graph for the Common Threshold and many of the same reasons apply here as well. The "no words" list also produces too many similarities and results in a very lengthy filtration phase that returns virtually no useful results. The longest list of common words produces fewer similarities but unlike the case with the Common Threshold, increasing

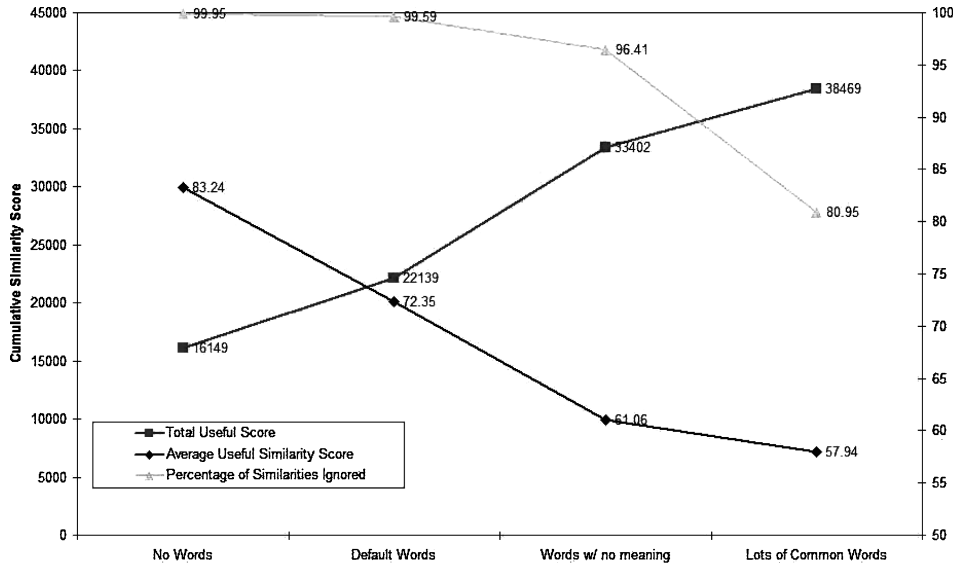

Fig. 8. Comparison of different common word lists.

the strictness meant the results were not meaningful, as they contained many similarities that do not make any sense when reading them. The results produced by the "words with no meaning" list were best since they retained the accuracy of the default list while having the advantage of not producing so many similarities, which saves time during detection and filtration.

4.2.7 *New Default Settings.* From the tests carried out it was clear that there were changes that could be made to the default settings in Sherlock that would increase detection speed, filtration speed, and the accuracy of results. These changes also serve to make Sherlock scalable to larger data sets such as A and B1. The new settings are:

Similarity Threshold	80
Common Threshold	8
Common Words	the, an, and, a , as, or, of, to
Maximum Links	6

These are significantly stricter than the original default settings. They achieve the desired results of more manageable similarity amounts for large data sets as well as improving accuracy and execution speeds. The Common Threshold and Common Words list have the most impact on the amount of similarities produced by the detection engine and increasing their strictness improved the quality of filtered results.

Subsequent use of these settings has been successful. However, there can be no such thing as an "all-purpose" group of settings for every possible size of data set. These settings are tuned to work for reasonably large classes on assignments of around 3000 words in length; smaller class sizes or analyzing

source-code comments will require different settings to achieve the same level of accuracy.

5. COMPARISON TO OTHER TOOLS

As part of the overall evaluation of Sherlock's new features, a comparison was undertaken with two other plagiarism detection tools that were also specifically designed to detect *intracorpal plagiarism* [Culwin and Lancaster 2001]. The two tools were CopyCatch Gold ES [Woolls 2004], and PRAISE (Plotted Ring of Analysed Information for Similarity Exploration) with VAST (Visualisation and Analysis of Similarity Tool) [Culwin and Lancaster 2004]. CopyCatch Gold ES is commercial software designed to work with electronic submission systems, such as BOSS [Joy and Luck 1998], that allow a tutor to download all the submissions for an assessment into one directory. The software is available free to higher education institutions in the UK. Its website claims it is in use by upward of 42 institutions, most of which are in the UK. PRAISE and VAST were developed at London South Bank University and are available from the website [Culwin and Lancaster 2004]. They are not production quality tools but research has been published on them [Culwin and Lancaster 2001].

CopyCatch is unique in this area as its algorithm was not designed by a computer scientist but rather by someone approaching the field from a linguist's point of view. The software searches for phrases, which are composed of a "function" word followed by either all the "content" words until the end of the sentence or another function word [Woolls 2003]. The exact algorithm used by CopyCatch is unknown due to the proprietary nature of the software and so it is therefore only possible to compare the results produced to other software rather than draw conclusions about how the algorithm may have produced better or worse results. From using the software it would appear that the class of function words is a superset of the closed-class words that Sherlock uses.

PRAISE is a tool for analyzing a set of submissions. It draws its results as a graph with submissions as nodes and similarities as weighted edges in a similar manner to the method Sherlock uses for its source-code results. Analyzing an edge will launch VAST, which generates an image representing the *fragmentary intercept similarity score matrix* and allows the user to highlight an area of the image. This highlighted area is represented in the original texts as different colored text. The algorithm used by PRAISE is to assign scores based on how many chains of a given length of words, sentences, or characters occur when comparing the texts. VAST's image generation works by breaking a document into *fragments*. Each pixel in the image is then colored on a grey-scale depending on the number of words two fragments have in common.

5.1 Quality of Results

The tools were then compared for the quality of their results and how well the results generated matched each other. For this test, detection was run on data set B1 as this set contained at least four plagiarism pairs according to the results from Sherlock. Here are details of the four pairs found by doing an

analysis of the results in Sherlock, in the order they were presented:

- *Pair* 1: Widespread verbatim copying, same grammatical errors, minimal rephrasing in other sections, no reordering of ideas.
- *Pair* 2: Some verbatim copying, same grammatical errors, some rephrasing, some reordering of ideas.
- *Pair* 3: Plenty of rephrasing and reordering, same grammatical errors. A third party also involved but to a lesser extent.
- *Pair* 4: One passage where both have virtually the same wording. Could be plagiarizing the same source.

In the first three pairs the certainty of collusion is because the pairs share the same errors, which, when coupled with the extensive similarities found throughout these pairs, makes the case for plagiarism very strong. The fourth pair have few similarities other than a long passage where they are virtually identical, suggesting that they have not colluded but have plagiarized the same source.

Next, detection was run using PRAISE and VAST. Pair 2 was given top ranking by PRAISE, while Pair 1 was placed second. Pair 3 was ranked third and the other document that was similar to this pair could also be found after a little investigation. The fourth pair was not ranked highly and proved difficult to find. In addition, one more pair worthy of investigation was identified:

- *Pair* 5: Some verbatim copying, possibly from the same sources. Some paraphrasing.

This pair was recognized by Sherlock but because of the low occurrence of similarities was not given a very high score. Even though the PRAISE results were quite similar to those generated by Sherlock, it was quite hard to find exactly which areas of the document were similar due to the nature of the images generated by VAST. This is because the image only points to general areas of the documents that are worth investigating rather than giving exact sentences.

Finally, detection was run using CopyCatch. Pair 1 was placed 14th on the similarity list generated by CopyCatch, while Pair 2 was given top ranking. Pair 3 was ranked 15th, while Pair 4 did not feature highly in the list, as was Pair 5.

The similarity score generated by CopyCatch seemed inconsistent in some places. An example of this occurred with many highly ranked pairs, where the amount of similarity between documents was very low but appeared to be considered worthy of investigation. An example of this was the pair ranked 6th by CopyCatch, which only had three sentences marked as being similar. Two other pairs found by CopyCatch were considered worthy of investigation:

- *Pair* 6: Very similar introductions, one essay cited a guest lecture given at the university while the other appeared to be using the same material without citation.
- *Pair* 7: Nearly identical conclusions.

Both these pairs were found by Sherlock and PRAISE but did not feature prominently and would probably have gone un-investigated. The pairs from four through to seven were all lesser instances of plagiarism than the top three and were probably more attributable to either a bad understanding of how and when to cite sources or laziness. The top three pairs were all considered to be definite instances of plagiarism worthy of investigation as they showed definite collusion and a disregard for the university's rules on collusion in assessed work.

5.2 Visualization of Results

As has already been mentioned, in the VAST results visualization was considered to be vague when the exact locations of suspicious were required. With some high-scoring pairs, it was also hard to see any areas of overlap between the documents as the whole image was composed of very slightly varying shades of grey.

The visualization results in CopyCatch is quite similar to the method used by Sherlock. Pairs were presented in order of calculated similarity and then individual pairs could be examined with suspicious text highlighted. The major problem with this was the lack of any way to quickly link between similar sentences by clicking on them. The similar sentence had to be found by scrolling the opposite pane to the given paragraph and sentence. Both CopyCatch and Sherlock support exporting results to a HTML format for external viewing.

Sherlock presented a user-friendly visualization; the hypertext metaphor is now well understood by all reasonably computer-literate people. User feedback during the development of the software indicated that the interface allowed rapid navigation around documents and examining similarities by clicking on sentences encouraged users to investigate while the familiar metaphor meant the learning curve was very shallow. In addition, having a sentence highlighted as the user moved the mouse across it meant it was clear which sections of the text were considered to be similar.

5.3 Speed of Detection

When comparing the different tools for detection speed, they were all run on the same machine (650 MHz Athlon, 384 MB RAM, Java 1.4, Windows XP), using the same data set with the same background programs loaded. The essays from set B1 were used, which consisted of 95 essays with a recommended length of 3000 words.

Sherlock Detection (sec)	1407
PRAISE Detection (sec)	408
CopyCatch Detection (sec)	25

From these results it is certainly obvious that the commercial program has a definite speed advantage. When considering the time for PRAISE it is also worth noting that the act of generating an image to view the similarities in VAST also takes an additional 87 sec for each pair viewed, while viewing results in CopyCatch is instantaneous and Sherlock takes 12 sec to load its results

and then all similarities can be viewed instantaneously. All three tools were written in Java so there was no possibility that one gained speed through native compilation.

6. EXPERIENCES USING SHERLOCK

While the Sherlock implementation of our algorithm should still be considered nothing more than a proof of concept, the tool has been successfully used on courses within the department. Instances of suspected collusion between students were identified, leading to investigations by academics. While we cannot discuss specific cases due to confidentiality issues, it suffices to say that the tool has proved useful.

With a prototype tool such as this, there are still many limitations on its use that would not exist in a more polished version: Sherlock currently has no ability to process documents that are not given in a plain text file. This also precludes the use of HTML documents. However, given the open-source nature of the software and the availability of open-source file conversion libraries, it would be possible to add such features to the software.

It is our experience that a clear submission policy is required for students to adhere to. On many courses within the department, students are asked to submit an electronic copy of their work via our BOSS Online Submission System [Joy and Luck 1998], but a specific format is not required of them. This leads to a variety of document formats, ranging from Microsoft Word to Adobe PDF to LaTeX, which further complicates the processing task. Having to convert all essays to plain text files is currently a very time consuming task and prevents the widespread use of the software within the department.

Sherlock is available as part of the BOSS Online Submission System (BOSS), an online system for the submission and marking of student work. Both BOSS and Sherlock are licensed under the GPL and so are freely available along with their source code from http://sherlock.org.uk/.

Many universities in the UK now provide access to the Joint Information Systems Committee (JISC) plagiarism detection service as a means of checking for direct plagiarism from websites and digital libraries [iParadigms 2005]. This is a fast and easy way to check large bodies of work for plagiarism. Sherlock's advantage over such a service is that it is better able to detect instances of paraphrasing or rewording. The standalone nature of Sherlock also lessens the issues with data protection laws as there is no need to transmit a student's work to a remote server for plagiarism detection. This advantage also applies to the other software Sherlock was compared to.

7. CONCLUSIONS

We have designed an algorithm for identifying similarities in collections of documents and have implemented and tested the algorithm on several large data sets. Comparing documents at a sentence-based level is a novel approach to the problem and we believe it offers new possibilities for plagiarism detection. The sentence-based approach allows for a more useful comparison than simply looking for chains of words.

While the chosen method for filtering the results may appear oversimplistic, the results obtained are very useful. Future work could take place in improving the filter by making it more sophisticated, perhaps by only keeping a proportion of sentences such that they contain lower numbers of similarities. Another method would be to make the preprocessor more intelligent by eliminating sentences that occur within quotations or have a citation at the end. However, there are many issues with deciding where the citation applies to or even whether the citation is sufficient to prevent a charge of plagiarism. Such issues are discussed extensively in Chapter 4 of [Decoo 2002].

Comparison with other natural language plagiarism detection tools indicates that the algorithm is accurate for detecting clusters of similar documents and generated similar results, although it was considerably slower, which may be due to the fact that the software is still at the prototype stage and the code has not yet been optimized.

Finally, use of the software by academics has indicated that Sherlock's visualization results is a good model to use as it makes users feel comfortable thanks to the familiar hypertext metaphor.

7.1 Future Work

An immediate improvement is to further optimize the software so that detection speeds approach those of commercially available software, such as CopyCatch. A further improvement would be an archive of submitted files that it could then check against new submissions. This would allow it to catch plagiarism across different years or from different courses. While the current implementation would be far too slow for such a proposal, this could be countered by using the fingerprinting techniques outlined in Hoad and Zobel [2003] and Finkel et al. [2002] as a way of determining which documents merit comparison using the thorough sentence-based approach. Further testing of the parameters for different types of data sets, such as comments from source code and smaller comparison sets, would also be desirable.

REFERENCES

CARROLL, J. AND APPLETON, J. 2001. Plagiarism: A good practice guide. Tech. rep., Joint Information Services Committee. Available: http://www.jisc.ac.uk/index.cfm?name=project_plag_practise (Accessed 27th January 2004).

CHESTER, G. 2001. Pilot of free-text detection software. Tech. rep., Joint Information Services Committee. Available: http://www.jisc.ac.uk/index.cfm?name=project_plag_pilot (Accessed 20th April 2005).

CULWIN, F. AND LANCASTER, T. 2001. Visualising intra-corpal plagiarism. In *5th International Conference of Information Visualisation (IV 2001)*. London, England.

CULWIN, F. AND LANCASTER, T. 2004. Plagiarism prevention and detection. online. Available: http://cise.lsbu.ac.uk (Accessed 20th April 2005).

CULWIN, F., MACLEOD, A., AND LANCASTER, T. 2001. Source-code plagiarism in uk he computing schools. In *2nd Annual Conference of the LTSN Centre for Information and Computer Sciences*. University of North London, England.

CURTIS, P. 2003. Hodge defends higher education target. online. Available: http://education.guardian.co.uk/print/0,3858,4582592-108229,00.html (Accessed 20th April 2005).

DECOO, W. 2002. *Crisis on Campus: Confronting Academic Misconduct*. The MIT Press, Cambridge, MA.

FINKEL, R. A., ZASLAVSKY, A., MONOSTORI, K., AND SCHMIDT, H. 2002. Signature extraction for overlap detection in documents. In *Proceedings of the 25th Australasian Conference on Computer Science*. Australian Computer Society, Inc., 59–64.

HOAD, T. C. AND ZOBEL, J. 2003. Methods for identifying versioned and plagiarized documents. *Journal of the American Society for Information Science and Technology 54*, 3, 203–215.

IPARADIGMS. 2005. Jisc service—solutions for a new era in education. online. Available: http://www.submit.ac.uk (Accessed 20th April 2005).

JOY, M. S. AND LUCK, M. 1998. *Computer Based Assessment (Vol. 2): Case Studies in Science and Computing*. *SEED Publications, University of Plymouth, United Kingdom*. The BOSS system for on-line submission and assessment of computing assignments, 39–44.

JOY, M. S. AND LUCK, M. 1999. Plagiarism in programming assignments. *IEEE Transactions on Education 42*, 2, 129–133.

MONOSTORI, K., ZASLAVSKY, A., AND BIA, A. 2001. Using the matchdetectreveal system for comparative analysis of texts. In *Proceedings of the 6th Australian Document Computing Symposium (ADCS 2001)*. 51–58.

MONOSTORI, K., FINKEL, R. A., ZASLAVSKY, A., HODASZ, G., AND PATAKI, M. 2002. Comparison of overlap detection techniques. In *International Conference on Computational Science (ICCS 2002)*. Amsterdam, The Netherlands, 51–60.

PRECHELT, L., MALPOHL, G., AND PHILLIPSEN, M. 2002. Finding plagiarisms among a set of programs with jplag. *Journal of Universal Computer Science 8*, 11.

RIBLER, R. L. AND ABRAMS, M. 2000. Using visualization to detect plagiarism in computer science classes. In *IEEE Symposium on Information Visualisation*. Salt Lake City, Utah, 173–177.

WITTEN, I. H., MOFFAT, A., AND BELL, T. C. 1999. *Managing Gigabytes: Compressing and Indexing Documents and Images*, 2nd Edn. Morgan Kaufmann, San Francisco, California.

WOOLLS, D. 2003. Private correspondence.

WOOLLS, D. 2004. Welcome to the home of powerful text analysis tools. online. Available: http://www.copycatchgold.com/ (Accessed 20th April 2005).

Received February 12, 2004; revised April 20, 2005; accepted April 25, 2005