CLASSIFICATIONS OF PLAGIARISM DETECTION ENGINES

Thomas Lancaster,
Business School,
University of Central England,
Perry Barr,
Birmingham B42 2SU,
UK.
thomas.lancaster@uce.ac.uk

Fintan Culwin,
Centre for Interactive Systems Engineering,
School of Computing,
London South Bank University
101 - 103 Borough Road,
London SE1 0AA
UK
fintan@lsbu.ac.uk

CONTENTS

ABSTRACT

Plagiarism detection engines are programs that compare documents with possible sources in order to identify similarity and so discover student submissions that might be plagiarised. They are currently classified as either attribute counting systems or structure metric systems. However these classifications are

inconsistently applied and inadequate, an area of particular relevance when classifying free text detection engines as this is where most current research is focused.

This paper proposes a new and alternative set of classifications based primarily around the types of the metrics the engines use. Current detection engines are classified using the new groupings. These new classifications are intended to allow detection engines to be discussed and compared without ambiguity.

1. INTRODUCTION

Student plagiarism, the process where students submit work for academic credit that contains other people's unacknowledged words or ideas, should be a cause for concern for all academic institutions. Numerous authors have assessed the problem, provided advice or stated that plagiarism may be endemic, these include Buckell (2002) and Culwin & Lancaster (2001a, 2001b). A particular concern of *Web plagiarism* has been identified, where students 'copy and paste' work from Internet sites; authors focusing on this type of plagiarism include Austin & Brown (1999), Culwin & Lancaster (2000b) and Lathrop & Foss (2000).

Although recommended, writing *assignment specifications* that limit plagiarism is only a partial cure. Some students will continue plagiarise regardless of how hard tutors try to stop them. As such *plagiarism detection systems*, those partially automated tools intended to find plagiarism in *student submissions*, are becoming increasingly necessary. These might be used primarily as deterrents or to preserve academic integrity. The software that compares student submissions with each other or with potential sources are known as *plagiarism detection engines*. Such engines may form one component of a larger detection system. Engines that operate over the Web in order to identify sources for a document are included within this definition of a detection engine, however the process by which they then present commonality to a user is not. This is to ensure that the process of detection, i.e. applying the engine, is separated from any process through which a student may be judged to be a plagiariser.

In their most traditional and basic form detection engines date back over twenty years to the ideas of Halstead (1977) and Ottenstein (1977). The traditional classifications used with these engines and subsequent engines were intended to classify only detection in program source code submissions, such as that used in programming assignments, not in free text submissions, such as student essays. This paper presents a number of alternative ways in which detection engines can be classified. The new classifications are proposed to be improvements over the traditional classifications.

The classifications are based on those presented by Lancaster (2003) in his PhD thesis. More technical details of how they could be applied purely to source code detection engines are available (Lancaster & Culwin 2004). The language used is consistent with that presented by Culwin and Lancaster (2000a) in

their plagiarism taxonomy, designed to allow plagiarism and associated issues to be discussed. Throughout this paper terms that are introduced for the first time are presented in italics. They are also defined in the glossary at the end of the paper.

2. DESCRIBING PLAGIARISM

In order to describe plagiarism detection engines it is necessary to catalogue the types of academic misconduct that they are intended to detect. This short taxonomy is a subset of that presented by Culwin and Lancaster (2000a).

A collection of submitted work is known as a *corpus*. Where the source and copy documents are both within a corpus this is known as *intra-corpal plagiarism*, or occasionally as collusion. Where the copy is inside the corpus and the source outside, for instance in a textbook, a submission from a student who took the assessment in a previous session, or on the Web, this is known as *extra-corpal plagiarism*.

An important differential between *source code plagiarism* and *free text plagiarism* is that the methods used to detect both of these differ. Source code detection is a well-understood area that has not recently been the focus of much research. It is thought to be easier to detect source code plagiarism than free text plagiarism since the language that can be used is constrained to a set of defined key words and since any plagiarism is most likely intra-corpal in nature. Free text plagiarism contains an effectively unlimited number of possible words that can be used and plagiarism may be intra or extra-corpal. Research on detecting plagiarism in free text is more recent and ongoing and has become possible due to the increasing availability of cheap computer processing power.

Detection engines are not limited to operating on free text or source code. They may be used to find similarity in spreadsheets, diagrams, scientific experiments, music, or any number of non-textual corpora. Although the classifications that will be presented in this paper are broadly applicable to any area it is not possible to specify all possibilities for non-textual documents so these classifications will primarily be applied to source code or free text.

3. TRADITIONAL CLASSIFICATIONS

As part of his PhD thesis Lancaster (2003) carried out a comprehensive study of the plagiarism detection engine literature, of which the listed classifications are discussed here. No sources were identified that applied these classifications to free text detection engines. This might be because the traditional classifications were only intended for source code engines, as free text engines were not available during this development. The literature was found to classify engines into two types, *attribute counting systems*

and *structure metric systems*. This section will demonstrate how recent definitions of these classifications have been both inconsistently applied. They are also immediately limited, since there are engines in existence that do not fit within either of these classifications.

The existing definitions are believed to be most useful from a chronological viewpoint, since early engines were attribute counting in nature. Later these were followed by structure metric systems that largely superseded the attribute counting systems. Within these contexts a *metric* can be thought of as the rule used to convert one or more submissions into a numeric value, representing its similarity. The numeric measurement of how similar two submissions are will be known as a *similarity score*.

Many of the source code plagiarism detection engines use a pre-processing stage known as *tokenisation*. This is where different parts of the code are replaced by a consistent token, for instance all keywords representing different types of loops in a corpus are replaced by the same token regardless of their original type. Tokenisation could be applicable to free text, for instance by replacing all nouns with a common token, but the technique is not known to be used by any existing free text detection engines.

The definitions presented in the literature are of interest. Verco and Wise (1996a, 1996b) report what they state as the common classifications, attribute counting systems and structure metric systems. Attribute counting systems are presented as those that measure some property of an individual system, known loosely as an *attribute count*. Examples given include the number of operand occurences in a program or a count of the number of possible different paths through a program. Structure metric systems are presented as any engines that look for similarity in a representation of two pieces of source code in addition to using the attribute counting techniques above; the techniques used will be referred to as *structure metrics*. Verco and Wise acknowledge that their own YAP3 detection engine, a structure metric system, does not use attribute counts. This presents an immediate shortcoming of the classifications since the group that contains YAP3 cannot be differentiated from an engine that uses structure techniques but not attribute counts.

Culwin, MacLeod and Lancaster (2001a) provide definitions vaguely in line with those of Verco and Wise. Their definition of attribute counting systems is further constrained by describing only those engines that require superficial examination of code with no knowledge of the structure or linguistical features. Structure metric systems are said to only use tokenisation techniques and the representative tokens searched for long common substrings. This is immediately inconsistent with Verco and Wise (1996a, 1996b). For instance, an engine using both attribute counts and structure metrics would be classified by Verco and Wise as a structure metric system, whilst it would not fit into the classifications given by Culwin, MacLeod and Lancaster. More recently Jones (2001a, 2001b) failed to differentiate between the two classifications, stating that his attribute counting system works along the same principles as the YAP3 structure metric

system. This shows there is confusion amongst even developers of detection engines. The classifications cannot be used to differentiate between engines that do or do not use tokenisation techniques.

It would be possible to redefine the existing classifications more formally to make them more consistent but the lack of consistent application is not the only problem. More crucially there are source code detection engines that cannot be classified as either attribute counting systems or structure metric systems under the definitions used by Verco and Wise, Culwin, MacLeod and Lancaster or Jones. One example is the prototyped detection engine by Saxon (2000) that finds similarity based on the compressibility of source code. This uses neither structure metrics nor attribute counts. Hence it could not easily be classified as either.

There are also problems with using these classifications to discuss free text detection systems. The tokenisation stage that is considered essential for structure metric systems by Culwin, MacLeod and Lancaster is not really defined for free text, since tokens have not been formally defined in this context. Further the names of the classifications are strongly geared towards source code. No literature has been identified where words and characters are considered to be attributes, two examples that one would imagine could be immediately comparable. Further structure within the confines of a sentence or other collection of words meeting layout properties of the English language is very different to that definition that could be considered for structure of program source code.

Hence, although the existing classifications can be considered useful from a historical viewpoint and to a lesser extent to compare existing source code detection engines, they are not considered suitable for free text detection and this is where the majority of recent research lies. This paper proposes a new set of more inclusive classifications along with ways to describe detection engines that are applicable to both source code and free text engines. The engines will be most generally classified by the corpora they can operate on and more specifically by the metrics that they use. A new and more inclusive set of metrics will be introduced to make this possible.

4. CLASSIFICATIONS BY THE TYPES OF CORPORA PROCESSED BY THE ENGINE

Some existing features of detection engines have already been described and these can be mapped directly onto classifications. Although academically unexciting it is essential to know whether engines operate on source code, free text or both. Similarly it is necessary to know whether they operate intra-corpally, extra-corpally or both. One classification largely missed in the literature but necessary for complete classification coverage is know whether engines operate on text or not. Hence they can be classified as textual or non-textual. An example of a non-textual classification would be an engine that aimed to find plagiarism of

design notation diagrams, although it must be stated that this is a possible type of engine for which there are no known examples.

The languages which the detection engines operate on are also of interest. For source code engines these are the programming languages, such as C, Java or Prolog. For free text engines these are the foreign languages, such as English, Chinese or Russian. Clearly engines may work on more than one of these languages but it is likely that the most effective techniques for finding similarity will be language dependent, based perhaps on linguistic properties of the chosen language. For source code engines it is also necessary to know whether the code must be parsable or not, as some detection engines do not work on unfinished student submissions. It is also necessary to know whether the engine works on tokenised or untokenised documents.

5. CLASSIFICATIONS BY THE AVAILABILITY OF THE ENGINE

It is useful to know where an engine operates, if it is on a *local* machine or if it *Web-based*. This refers to where the corpus of student submissions is processed, whether it is on a local machine or if the files need to be uploaded and processed remotely. Some locally operating engines might use Web sources, for instance free text engines that search the Web, but these would still be classified as local, since the submissions are processed on the home machine.

It is also useful to know the availability of an engine. Those available to anyone are classified as *public* whilst those available only to their host institution are known as *private*. Those that can be made available by request are known as *special arrangement*, perhaps through supplying submissions directly to the engine provider. This latter category includes those engines that are available only after paying a charge or subscription fee, such as many of the Web-based detection services. Since not all engines described in the literature are still available they can also be classified as *past* or *current*, where past denotes those engines that are no longer in use and current denotes those that are still used.

6. CLASSIFICATIONS BY THE NUMBER OF SUBMISSIONS PROCESSED BY THE METRICS USED

The methods that the detection engines used to find similarity are of most academic interest. It is suggested that metrics can be differentiated from one another based on the number of documents that are processed together to generate them, a set of classifications not previously found in the literature. Studying technical descriptions of detection engines based on the documents they process two main types of metrics have been identified; it is proposed to name these *singular metrics* and *paired metrics*. These operate on one and two

documents at a time respectively to generate a numeric value. Paired metrics are intended to give more information that could be gleaned by simply computing and combining two singular metrics.

For completeness *corpal metrics* and *multi-dimensional metrics* are also defined here, each of which operates simultaneously on a greater number of documents. A corpal metric operates on an entire corpus at a time, for instance to find some general property of it. One use of this might be to compare the standard of work from different tutor groups. A multi-dimensional metric operates on a chosen number of submissions, so a singular metric would be 1-dimensional, a paired metric 2-dimensional and a corpal metric n-dimensional where n is the size of the corpus. Multi-dimensional metrics might be useful for finding clusters of similar submissions.

Table 1 contains some examples of possible metrics that fall under each classification. Examples are given for both source code and free text although some might prove to be inappropriate for detection.

	Source Code	**Free Text**
Singular Metrics	Mean number of characters per line. Proportion of 'while' loops to 'for' loops	Mean number of words per sentence. Proportion of use of 'their' compared to 'there'.
Paired Metrics	Number of keywords common to two source code submissions. The length of the longest tokenisation substring common to both.	Number of capitalised words common to two free text submissions. The length of the longest substring common to both.
Multi-Dimensional Metrics	The proportions of keywords common to a set of submissions.	The proportion of words from a chosen group common to a set of submissions.
Corpal Metrics	The proportion of source code submissions using the keyword 'while'.	The proportion of submissions using the word 'hence'.

TABLE 1 - Examples of Dimensional Metrics

7. CLASSIFICATIONS BY THE COMPLEXITY OF THE METRICS USED

A study of the existing technical literature has allowed two main classifications to be developed, based around the computational complexity of the methods employed to find similarities. These groups have been named *superficial metrics* and *structural metrics*.

A superficial metric is a measure of similarity that can be gauged simply by looking at one or more student submissions. No knowledge of the structure of a programming language or the linguistic features of natural language is necessary. A structural metric is a measure of similarity that requires knowledge of the structure of one or more documents. For source code submissions this might involve a parse of the submissions. For free text submissions this could involve reducing words to their linguistical root word form.

Table 2 gives some examples of superficial and structural metrics.

	Source Code	**Free Text**
Superficial Metrics	The count of the reserved keyword 'while'.	The number of runs of five words common to two submissions.
Structural Metrics	The number of operational paths through a program.	The size of the parse tree for a submission.

TABLE 2 - Examples of Operational Complexity Metrics

Although these categories are fully inclusive and mutually exclusive it is impossible to give a definition that can be consistently applied in every case. The borderline between where a superficial metric stops and a structural metric begins is necessarily a fuzzy one. For instance if a submission is tokenised and a superficial metric applied the whole process could instead be thought of as just a structural metric, since tokenisation is a structure dependent process. Hence in some cases these definitions are open to individual interpretation.

Of further academic interest is the operational complexity for the metrics used. Roughly this is the run time proportional to the size of the corpus. This could be provided as optional additional information when describing metrics. Most intra-corpal plagiarism detection engines work by comparing every submission with every other possible submission, giving time complexity proportional to the square of the number of submissions (known as $O(n^2)$ for n submissions). This means that processing time increases exponentially as the number of submissions grows. More computationally efficient comparison methods may take less time, an issue which is important when considering scalability, for instance when a free-text engine is being linked to a sizeable database of possible sources.

8. CLASSIFICATIONS OF CURRENT DETECTION ENGINES

There are a number of detection engines for both free text and source code that are still in use. Table 3 shows some of the major engines that have been identified. Each is described based on the new proposed

classifications as far as technical information can be gleaned from the literature. Since some literature descriptions are incomplete it is impossible to guarantee the accuracy of all the classifications. Where necessary intelligent guesses have been made. Jones (2001), Saxon (2000) and Tetlow (2004) do not name their engines so they are referred to here using names of the authors. There are also cases where engines are described under multiple names. The iParadigms product TurnItIn.com appears to have been most commonly referred to in the literature under the company name and is hence also referred to her in that manner. To further complicate matters the service has been made available in the UK as the JISC plagiarism detection service (JISC 2004). The free text detection engines are all listed as operating in English only, since no mention of their usefulness on other languages is known, although it is expected that they would be applicable to other Westernised languages, since the language structure can be considered to be broadly similar. For each engine a reference has been included in the table so that further information can be gleaned on it.

													Metrics			
Engine name	**Source code**	**Free text**	**Intra-corpal**	**Extra-corpal**	**Local**	**Web-based**	**Public**	**Private**	**Special arrangement**	**Past**	**Current**	**Tokenisaiton**	**Singular**	**Paired**	**Structural**	**Superficial**
Big Brother	√	√	√		√				√		√	√		√	√	
CopyCatch		√	√		√				√		√			√		√
CopyFind		√	√		√				√		√			√	√	
DetectaCopius	√		√		√		√				√			√		√
EduTie		√	√	√		√			√		√			√	√	
EVE2		√		√	√				√		√			√		√
Ferret		√			√		√				√			√		√
Gossip		√	√	√	√				√		√			√		√
iParadigms		√	√	√		√			√		√			√	√	

(aka TurnItIn.com)																
Jones	√		√		√			√			√	√	√			√
JPlag	√		√			√				√		√		√	√	
MOSS	√		√			√					√	√		√	√	
MyDropBox (formerly PlagiServe)		√		√		√	√				√			√	√	
Saxon	√		√		√				√		√			√	√	
SHERLOCK	√	√	√				√				√	√		√	√	√
Tetlow	√		√		√				√		√	√		√	√	
TRANKER		√	√		√				√		√			√		√
WORDCheck		√	√		√				√		√			√		√

TABLE 3 - Classifications of Current Detection Engines

9. CONCLUSIONS

The number of available solutions for plagiarism detection, especially for free text, is continually growing, but there has been no consistent way of organising or describing these engines. The existing classifications for source code detection engines have been shown to be inconsistently applied and are not obviously applicable for free text engines.

This paper has argued that a new set of classifications would be more suitable. Engines can most simply be described based on the corpora they operate on and their availability. They can be described more specifically on the underlying metrics they use to find similarity. In particular, it is useful to know the number of submissions that are processed at a time and the complexity of the metrics used.

It is intended that these new inclusive classifications allow detection engines to be discussed and compared more accurately in the literature so that effective solutions to student plagiarism can be identified. Existing current detection engines have been tabulated using these new classifications. These detection engines should be used in conjunction with a suitable pro-active anti-plagiarism policy to ensure that the value of academic awards is preserved.

10. GLOSSARY

Assignment specifications - The description of a task that students must carry out successfully to gain academic credit.

Attribute counting systems - Traditional definition of a detection engine using only attribute counts.

Attribute counts - Traditional description of a count of some property of a single document.

Corpal metrics - A metric where every document within a corpus must be processed together to give a numeric representation of that corpus.

Corpus - A collection of documents, in the plagiarism detection the student work to be checked and possibly the potential sources.

Current engine - An engine in use at the present time.

Extra-corpal plagiarism - Plagiarism where the copy is inside a corpus and the source is outside it.

Free text plagiarism - Plagiarism involving natural language, such as student essays.

Intra-corpal plagiarism - Plagiarism where the source and copy are both inside a corpus.

Multi-dimensional metrics - An n-dimensional metric is one where n documents in a corpus must be processed together to give a numerical representation.

Paired metrics - A metric where two documents are processed together to give a numerical representation.

Past engine - An engine that is no longer believed to be in use.

Plagiarism detection engines - The software that compares a corpus of student submissions with potential sources to find similarity.

Plagiarism detection systems - An automated or partly automated tool that can to use to aid tutors in the plagiarism detection process.

Private engine - An engine that is used only within its host institution.

Public engine - An engine that is made available for anyone who wishes to use it.

Similarity score - A numeric measure of the similarity in a pair of documents.

Singular metrics - A metric where a single document is processed to give a numeric representation of it. Multiple singular metrics can be combined using a closeness calculation to discover how similar documents are.

Source code plagiarism - Plagiarism of computer programs.

Special arrangement engine - An engine that can be made available only by negotiation with its owner.

Structural metric - A metric that measures a property of a document or a number of documents where knowledge of their structure is necessary.

Structure metric systems - Traditional definition of a detection engine using structure metrics and possibly attribute counts. Found to be inconsistently applied.

Structure metrics - Traditional description of a count of a property of a pair of documents that might involve tokenisation. Found to be inconsistently applied.

Student plagiarism - the use of the words or ideas of another without acknowledgement for academic credit.

Superficial metrics - A metric that can produce a numerical representation of a document or set of documents where no knowledge of their linguistic properties is necessary.

Tokenisation - A pre-processing detection stage where terms are replaced by representative symbols.

Web plagiarism - Extra-corpal plagiarism where the source is outside a corpus.

Web-based engine - An engine that is available for use over the Web. Documents may be processed on a remote server.

REFERENCES

Austin, M. & Brown, L. (1999) Internet Plagiarism: Developing Strategies to Curb Student Academic Dishonesty, *The Internet and Higher Education*, **2**, 1, 21-33.

Buckell, J. (2002) Plagiarism Tracked at 8 Per Cent, Australian IT,11/09/02, Available online at http://australianit.news.com.au/articles/0,7204,5071781%5e15334%5e%5enbv%5e15306-15317,00.html.

Boywer, K. W. & Hall, L. O. (1999) Experience using 'MOSS' to Detect Cheating on Programming Assignments, *29th ASEE/IEEE Frontiers in Education Conference*, San Juan, Puerto Rico, pp18-22.

Braumoeller, B. & Graines, B. (2001) Actions Do Speak Louder Than Words: Deterring Plagiarism with the use of Plagiarism Detection Software, available online at http://www.apsanet.org.PS/dec01/braumoeller.cfm.

Bull, J., Collins, C., Coughlin, E. & Sharp, D. (2001) Technical Review of Plagiarism Detection Software Report, Joint Information Systems Committee, available online at http://www.jisc.ac.uk/plagiarism..

Chester, J. (2001) Pilot of Free-Text Electronic Plagiarism Detection Software, Joint Information Systems Committee, available online at http://www.jisc.ac.uk/plagiarism.

CopyFind (2004) Available online at http://plagiarism.phys.virginia.edu/software.html.

Culwin, F. & Lancaster, T. (2000a) A Descriptive Taxonomy of Student Plagiarism, unpublished, available from South Bank University, London, UK.

Culwin, F. & Lancaster, T. (2000b) A Review of Electronic Services for Plagiarism Detection in Student Submissions, *Proceedings of 1st LTSN-ICS Conference*, Edinburgh, pp54-61.

Culwin F. & Lancaster T. (2001a) Plagiarism Issues for Higher Education, Vine 123, available from LITC, South Bank University, London.

Culwin F. & Lancaster T (2001b) Plagiarism Prevention, Deterrence & Detection, available online at http://www.ilt.ac.uk/resources/Culwin-Lancaster.htm.

Culwin, F. MacLeod, A. & Lancaster, T. (2001) Source Code Plagiarism in UK HE Computing Schools, Issues, Attitudes and Tools, *South Bank University Technical Report SBU-CISM-01-02*.

DetectaCopias (2004) available online at http://www.dcc.uchile.cl/~rmeza/proyectos/detectaCopias/index.html.

EduTie (2004) available online at http://www.edutie.com.

Halstead M. H. (1977) *Elements of Software Science*, Elseveir.

Irving, R., MacDonald, G., McGookin, D. & Prentice, J. (2002), *Big Brother (Glasgow University Computer Science Department's Collusion Detector System, version 2.0) User Manual*, available from Glasgow University.

JISC (2004) available online at http://www.jiscpas.ac.uk.

Jones, E. L, (2001a) Metrics Based Plagiarism Monitoring, *6th Annual CSSC Northeastern Conference, Middlebury, Vermont*, April 20-21 2001.

Jones, E. L. (2001b) Plagiarism Monitoring and Detection - Towards an Open Discussion, *7th Annual CSSC Central Plains Conference, Branson, Missouri*, April 6-7 2001.

Joy, Mike & Luck, Michael (1999) Plagiarism in Programming Assignments, *IEEE Transactions on Education*, **42**(2), pp129-133.

Lancaster, T. (2003) *Effective and Efficient Plagiarism Detection*, PhD thesis, available from London South Bank University, London, UK.

Lancaster, T. & Culwin, F. (2004), *A Comparison of Source Code Plagiarism Detection Engines.* To appear in Journal of Computer Science Education.

Lathrop, A. & Foss, K. (2000) *Student Cheating and Plagiarism in the Internet Era – A Wake Up Call*. Published by Libraries Unlimited Inc.

Lyon, C., Barrett, R. & Malcolm, J. (2004) A Theoretical Basis to the Automated Detection of Copying Between Texts, and its Practical Implementation in the Ferret Plagiarism and Collusion Detector, *Plagiarism: Prevention, Practice and Policy Conference, Newcastle, UK*, June 28-30 2004.

MyDropBox (2004) available online at http://www.mydropbox.com.

Ottenstein K. J. (1977) An Algorithmic Approach to the Detection and Prevention of Plagiarism, *SiGCSE Bulletin* **8**, 4, 30-41.

Prechelt, L., Guido, M. & Phlippsen, M. (2001) JPlag: Finding Plagiarisms Among a Set of Programs with JPlag, submitted to *Journal of Universal Computer Science*, Facultat fur Informatik, Universitat Karlsruhe, Germany.

Saxon, S. (2000) *Comparison of Plagiarism Detection Techniques Applied to Student Code*, Part II Computer Science Project, Trinity College, Cambridge, UK.

Tetlow, M. (2004) *Detecting Plagiarism in Visual Basic Source Code – A Comparative Investigation into Effective Plagiarism Detection Techniques*, Final Year Computing Project, Business School, University of Central England, Birmingham, UK.

Trott, J. (2004) *An Open Source Web Searching Tool to Detect Internet Plagiarism*, Final Year Computing Project, Business School, University of Central England, Birmingham, UK.

Verco, Kristina L. & Wise, Michael J. (1996a) Plagiarism a la Mode: A Comparison of Automated Systems for Detecting Suspected Plagiarism. *The Computer Journal* **39**, 9, 741-750.

Verco, K. L. & Wise, M. J. (1996b) Software for Detecting Suspected Plagiarism: Comparing Structure and Attribute-Counting Systems, *Proceedings of First Australian Conference on Computer Science Education*, July 3-5 1996, Sydney, Australia.