Old and new challenges in automatic plagiarism detection

Paul Clough, Department of Information Studies
(p.d.clough@sheffield.ac.uk)

University of Sheffield
Regent Court
211 Portobello Street
Sheffield, S1 4DP
UK

Automatic methods of measuring similarity between program code and natural language text pairs have been used for many years to assist humans in detecting plagiarism. For example, over the past thirty years or so, a vast number of approaches have been proposed for detecting likely plagiarism between programs written by Computer Science students. However, more recently, approaches to identifying similarities between natural language texts have been addressed, but given the ambiguity and complexity of natural over program languages, this task is very difficult. Automatic detection is gaining further interest from both the academic and commercial worlds given the ease with which texts can now be found, copied and rewritten. Following the recent increase in the popularity of on-line services offering plagiarism detection services and the increased publicity surrounding cases of plagiarism in academia and industry, this paper explores the nature of the plagiarism problem, and in particular summarise the approaches used so far for its detection. I focus on plagiarism detection in natural language, and discuss a number of methods I have used to measure text reuse. I end by suggesting a number of recommendations for further work in the field of automatic plagiarism detection.

1. Introduction

As an example of one particular type of text reuse (see, e.g. (Clough,2003) for further information and discussion), plagiarism has received much attention from both the academic and commercial communities. This has been particularly true in academia as students turn to technology to fabricate texts which are not entirely their own work. Text reuse is defined as the activity whereby pre-existing written material is reused during the creation of a new text, either intentionally or un-intentionally.

Many other examples of text reuse surround us today, including the creation of literary and historical texts, summarisation, translation or revision of existing texts. Many factors influence text reuse including translating an original text into a different language, restyling an original to fit different authorial or consumer needs (e.g. rewriting a scientific text to be readable by the layman), reducing or expanding the size of the original text and the competency and production requirements of the writer.

Recent advances in technology are making text reuse much easier. For example, the Google web search engine claims to index over 3 billion web pages[1] providing a large variety of source texts on a diverse range of topics in many different languages. Word processors have also become more sophisticated, enabling users to easily cut and paste, merge and format pre-existing texts from a variety of sources. This, coupled with the change of culture brought about by electronic 'cyber-space' has caused concern to authors surrounding the ownership of their written material. Either the owner (perhaps the publisher) has to protect their texts (e.g. using digital watermarks), rely on finding illegitimate copies, or even de-value[a] their digital content in some way. Mallon (1989) suggests that "the origin and ownership of all electronic documents is now peculiarly evanescent; one click of the 'Save As' button can give a whole new name and identity, instantly, to someone else's creation."

Of course not all text reuse is deemed a cardinal sin; there are certain areas where borrowing is perfectly acceptable. For example, as Angélil-Carter (2000:23) points out: "generally, borrowing is a tradition in literature and other art forms and more than a tradition: creativity feeds on what has gone before, new work is formed out of old." During the renaissance and romantic eras of literary writing, even the "great" authors would reuse the ideas, storylines and plots of others in their own literary creations. It was not considered immoral or unethical; rather it was seen as a stimulus for creativity. Text reuse was (and is) the epitome of literary recognition.

Today, there are examples where even verbatim text reuse is not considered unethical; rather the norm. One such example is the reuse of newswire (or news agency) texts by journalists in the creation of newspaper articles (see, e.g. (Clough et al.,2002a)). As long as the journalist or the organisation they write for are subscribers of the news agency, they are free to reuse agency text as and how they please, with or without attribution. But let us return to the broader example of text reuse in hand: plagiarism. Why is it considered so wrong and what can be done to counteract this kind of text reuse?

2. Plagiarism detection

The aim of this paper is to present plagiarism detection as a problem to be solved; not to cover other aspects of plagiarism, important as they are, such as: surrounding ethical and moral issues (see, e.g. (Mallon,1989), (Martin, 1994), (LaFollette,1992), (Hannabuss,2001) and (Angélil-Carter,2000)), suggestions for practical steps that the individual or institution can take to detect plagiarism (see,

[1] This estimate stands at the time of writing and not all of these web pages will contain text.

e.g. (Bull et al.,2001) and (Culwin and Lancaster,2000)), examples of plagiarism (see, e.g. (Mallon,1989)), reasoning behind student plagiarism (see, e.g. JISC), or guidance for writers on how to prevent themselves unintentionally plagiarising their sources. So what is plagiarism?

2.1. The problem of plagiarism

> *"The wrong in plagiarism lies in misrepresenting that a text originated from the person claiming to be its author when that person knows very well that it was derived from another source, knows that the reader is unlikely to know this, and hopes to benefit from the reader's ignorance." - (Samuelson,1994)*

Joy and Luck (1999) define plagiarism as "unacknowledged copying of documents or programs" that can "occur in many contexts: in industry a company may seek competitive advantage; in academia academics may seek to publish their research in advance of their colleagues." Most empirical study and analysis has been undertaken by the academic community to deal with student plagiarism, although methods of detection have found their way into the commercial world, e.g. measuring software reuse and identifying reused code (see, e.g. (Hislop,1998)).

Hannabuss (2001) defines plagiarism as the "unauthorised use or close imitation of the ideas and language/ expression of someone else and involves representing their work as your own." Plagiarism is closely linked with intellectual property and copyright, both of which are set in place to protect the ownership of texts through the creativity and originality of their contents. As Osen (1997) comments: "if plagiarism is the bane of the academic world, copyright infringement is the scourge of the legal one." Copyright law, however, does not protect ideas, themes or subject matter; only the style with which the content is expressed, i.e. it covers only form.

Plagiarism is considered a problem because it not only infringes upon existing ownership, but also deceives the reader and misrepresents the originality of the current author: "it is the mixture of law or rule-breaking and social misrepresentation and deception that gives plagiarism its bitter taste" (Hannabuss,2001). In education, students may plagiarise to gain a qualification; academics to gain popularity and status. If a plagiarism relationship exists between two texts, it suggests that the texts exhibit some degree of intertextuality, which would not appear between them if independently written.

Academia encourages students and researchers to build upon the ideas of others, but failure to acknowledge the ideas or material of others, or copying existing work is unacceptable. There are cases when what appears to be plagiarism is not, e.g. appropriate self-reuse of one's own work (see, e.g. (Samuelson,1994)), or poor citation. These cases can be resolved through manual inspection.

2.2. Forms of plagiarism

Plagiarism can take several distinct forms, including the following (Martin,1994):

(1) **Word-for-word plagiarism:** direct copying of phrases or passages from a published text without quotation or acknowledgement.

(2) **Paraphrasing plagiarism:** when words or syntax are changed (rewritten), but the source text can still be recognised.

(3) **Plagiarism of secondary sources:** when original sources are referenced or quoted, but obtained from a secondary source text without looking up the original.

(4) **Plagiarism of the form of a source:** the structure of an argument in a source is copied (verbatim or rewritten).

(5) **Plagiarism of ideas:** the reuse of an original thought[2] from a source text without dependence on the words or form of the source.

(6) **Plagiarism of authorship:** the direct case of putting your own name to someone else's work

The easiest form of plagiarism to detect and prove is verbatim or word-for-word text reuse (given a possible source text to compare with). This can often be detected using the simplest of automatic methods, but occurrences by students are often due to the fact that they are uncertain as to how to reuse source texts legitimately.

Other forms, such as paraphrasing and the reuse of structure can also be identified relatively easily, but get progressively harder as the plagiarist uses more complex rewrites or to hide the original text, or reuses only ideas and not the content. The extreme is *ghost-writing:* getting someone else to write the text for you. These forms of plagiarism are not just harder to detect, but also harder to prove.

2.3. Examples of text reuse

2.3.1 Student plagiarism

Real examples of student plagiarism are hard to come by due to restrictions in student confidentiality. However, many on-line sources exist that provide examples of what constitutes student plagiarism. The simplest form of plagiarism is to directly copy from a source text with minimum rewriting. This type of text reuse is common in student plagiarism (Martin,1994) where entire passages are copied word-for-word directly from the source. For example, consider the following rewrite from the University of Wisconsin-Madison Writing Centre[3]:

[2] There are cases when an idea or thought becomes common knowledge and it is not necessarily plagiarism if the idea is reused.
[3] http://www.wisc.edu/writing/Handbook/QuoSampleParaphrases.html (site visited 23/01/2003).

Original version:

"How important is our power of non-analytical thought to the practice of science? It's the most important thing we have, declares the Princeton physicist historian Thomas Kuhn who argues that major breakthroughs occur only after scientists finally concede that certain physical phenomena cannot be explained by extending the logic of old theories. Consider the belief that the sun and the planets move around the earth, which reigned prior to 1500. This idea served nicely for a number of centuries, but then became too cumbersome to describe the motions of heavenly bodies. So the Polish astronomer Copernicus invented a new reality that was based on a totally different 'paradigm' or model - that the earth and planets move around the sun".

Plagiarised version:

Non-analytic thought is considered very important to the **practice of science** by **Princeton physicist historian Thomas Kuhn** who claims **that major breakthroughs** happen only when **scientists finally concede** that some **physical phenomena** defy explanation by **extending the logic of old theories**. One idea which **served nicely** for many centuries but then **became too cumbersome** was **the belief that the sun and planets** revolved **around the earth**. This was held **prior to 1500** until **Copernicus invented a new reality: the earth and planets move around the sun**.

Words highlighted in bold are copied directly from the source and the structure follows a similar form to that of the original. Even with proper citation, there is little originality or added creativity in this example. Although text can be copied verbatim, it is more likely that sentences are rephrased or paraphrased to put the original into the writer's own words. The following examples are taken from the University of Kentucky[4]:

Original version:

"Those complexes that contain unpaired electrons are attracted into a magnetic field and are said to be paramagnetic, while those with no unpaired electrons are repelled by such a field and are called diamagnetic."

Plagiarised versions:

"**Complexes that contain unpaired electrons are** those that **are attracted** to **a magnetic field**. These **are** called **paramagnetic, while those with no unpaired electrons are repelled by a** magnetic **field and are** said to be **diamagnetic.**"

"**Those complexes that contain** paired **electrons are repelled by a magnetic field and are said to be diamagnetic**, whereas **those with no** paired **electrons are attracted** to such **a field and are called paramagnetic.**"

"Compounds **that** have **unpaired electrons are attracted** to **a magnetic field and are called paramagnetic.** Compounds **with no unpaired electrons are repelled by** this **field and are** said to be **diamagnetic.**"

The degree of rewriting can vary from direct copying from a source with no attribution, the insertion or deletion of grammatical units, e.g. sentences or phrases, the insertion or deletion of words within a sentence, e.g. noun phrase modifiers, the reordering of words in a sentence or the re-ordering of sentences in a discourse, inversion of original clauses, substitution of equivalent words or phrases, changes in form such as tense and voice (e.g. active to passive voice), making abstract ideas more concrete (specification), making concrete ideas more abstract (generalisation), merging or separating sentences (either adjacent or dispersed throughout the original) and rewriting direct quotes as indirect (and vice-versa). Rewriting a text will invariably include paraphrasing: a form of rewriting in which the meaning of the original phrase or sentence is preserved, but the way in which the meaning is expressed is changed.

Other forms of plagiarism include submitting someone else's work (or copying it directly), failing to reference/ footnote source material and copying a collection of paragraphs from a variety of electronic sources and pasting them together into a coherent whole. This is known as an 'Internet pastiche' or 'patchwork plagiarism'.

2.3.2 Journalism

An example of text reuse with which I have been involved more closely with lies within the domain of journalism. It is common knowledge within the newspaper industry that most newspapers rely heavily upon press agencies as their primary source of 'pre-fabricated' news material (Bell,1996:20-22). This is a form of plagiarism, but 'benign'. As regular subscribers to news agency services, media organisations pay an annual fee which entitles them to reuse the agency source text in any way they see fit, either verbatim, or editing it to suit their own production requirements. In many cases, the final attribution of the news story goes to the journalist or newspaper and not the news agency, but this is perfectly acceptable. As an example, consider the following.

Original (news agency):

A Chief Constable's daughter who assaulted two officers in her father's force after drinking a litre of strong cider was today sentenced to 150 hours community service.

Rewrite (The Sun - popular press):

A Top Cop's **daughter who assaulted two** of her Dad's **officers after** downing **a litre of cider was sentenced to 150 hours' community service** yesterday.

Rewrite (The Independent - quality press):

The **daughter** of the **Chief Constable** of Sussex **was sentenced to 150 hours' community service** yesterday.

The journalist may decide to rewrite, re-order, delete or paraphrase agency copy, rather than reuse the text verbatim depending on a wide variety of external influences and personal writing style. Texts are often edited after their creation by other newsworkers whose job it is to remove personal style and make it consistent with other texts written under the name of a common author (the newspaper). In the past, paper copies of stories were marked with these changes, but now journalists like other writers, work with entirely electronic media, which has promoted a cut-and-paste culture that makes it easier for the journalist and editor to create and manipulate versions of a story as it cycles between newsworkers prior to production. Bell (1991) identifies typical rewrite strategies

[4] http://www.chem.uky.edu/Courses/common/plagiarism.html (site visited 23/01/2003).

used by the newsworker as deletion, lexical substitution, and changes in syntax.

2.4. Plagiarism detection

Typically, manual plagiarism detection within a single text is based on identifying inconsistencies such as the author's writing style, or recognising passages with a familiar feel to them. Between multiple texts, plagiarism detection involves finding similarities which are more than just coincidence and more likely to be the result of copying or collaboration between multiple authors. In some cases, a single text is first read and certain characteristics found which suggest plagiarism. The second stage is to then find possible source texts using tools such as web search engines for unknown on-line sources, or manually finding non-digital material for known sources.

In academia, plagiarism detection is generally down to the knowledge, ability, time and effort of the lecturer/teacher assessing the student's work. Typical discriminators signaling plagiarism might include the following:

- Use of advanced or technical vocabulary beyond that expected of the writer.
- A large improvement in writing style compared to previous submitted work.
- Inconsistencies within the written text itself, e.g. changes in vocabulary, style or quality.
- Incoherent text where the flow is not consistent or smooth, which may signal that a passage has been cut-and-pasted from an existing electronic source.
- A large degree of similarity between the content of two or more submitted texts. This may include similarity of style as well as content.
- Shared spelling mistakes or errors between texts.
- Dangling references, e.g. a reference appears in the text, but not in the bibliography.
- Use of inconsistent referencing in the bibliography suggesting cut-and-paste.

In a recent survey undertaken by JISC (Bull,2001) in which 321 lecturers were asked to participate in a questionnaire regarding their plagiarism experiences, 72% of the respondents declared that a change in writing style within a text was enough to signal plagiarism.

From the literature surrounding plagiarism, I believe that at least four problems exist that those addressing plagiarism detection should address. I have classified these based on whether the detection addresses a single text or more than one text.

(1) **Within a single text:**

a) Identify inconsistencies that indicate a text is unlikely to be written solely by the claimed author.

b) Find the likely sources for an inconsistent text.

(2) **Between multiple texts:**

c) Identify unacceptable collaboration, i.e. collusion.

d) Identify unacceptable copying from a source text, i.e. plagiarism.

It is useful to make the distinction between identifying plagiarism within a single text and between multiple texts because much of the research in automatic plagiarism detection to date has concentrated on the latter. However, from the findings of the JISC survey and from the comments of lecturers at the University of Sheffield, it is likely that methods to address the first task are also important and necessary to cover all types of plagiarism.

2.5. What the plagiarism detection task is not

The problem of identifying plagiarism concerns the way in which the content of a text is expressed and whether this is likely to come from other sources. Is it consistent, or does it feel multi-authored; is the text similar to an existing one or does it appear original? This task differs from other problems of text analysis, for example authorship attribution and information retrieval.

In authorship attribution, the concern is one of authorship regardless of content. Discriminators of attribution tend to be those based on lexical or syntactic features which remain constant over texts on different topics, but vary across authors. Plagiarism detection also varies from the typical *ad hoc* information retrieval task of finding texts on the same or similar topic to a user-defined query. In this problem discriminators are those based on topic (e.g. perhaps noun phrases, or proper names).

The plagiarism detection task is different from authorship attribution, but deeper than information retrieval. Plagiarism concerns content, regardless of author and expression, rather than topic. Given that there will always remain a high degree of similarity between works on the same topic (e.g. essays written by a group of students on the same course) discriminators used in plagiarism detection should not only concern content, but also expression.

3. Automatic plagiarism detection

Almost all research to date has concentrated on identifying plagiarism and collusion *between* texts. Since the 1970s, the popularity and direction of automatic plagiarism detection has changed. To begin with, empirical research came from the programming community, particularly in academia where computer science departments built tools to identify "unusual" similarity between programming assignments handed in by students. There is still much work being done within industry where there is great interest in identifying similarity between large software programs, e.g. duplication, redundant code and similarity between revisions.

In academia, recent interest has shifted towards identifying plagiarism between natural language texts. Particular areas of concern include: identifying verbatim cut-and-paste (with minor changes) from Web-based sources and identifying

the same content but paraphrased. This is reflected by the increase in on-line services (e.g. Plagiarism.org and turnitin.com) to address plagiarism from available on-line resources, particularly term paper mills which can supply pre-made essays to students for a given fee. Services to track and monitor commercial content have also received increased popularity as the media report more cases of stolen digital content (e.g. contentguard.com).

Whale (1990) suggests that "the task [of automatic plagiarism detection] may be simplified by finding a distinctive characteristic such as a misspelled identifier or paraphrased comment, though such a capability is hard to build into any automated plagiarism detection system." To identify more complex forms of plagiarism beyond cut-and-paste or simple rewriting is hard, particularly for natural language, therefore automatic plagiarism detection generally involves finding quantifiable discriminators which can be used to measure the similarity between texts. Rather than identify more complex plagiarism, the automatic methods aim to highlight potentially derived texts through "unusual" or unlikely similarity enabling further manual inspection where more complex forms can be found.

Assumption:
The greater the similarity (or fewer the differences) between texts; the more likely is that one of the texts is derived.

One of many challenges in automatic plagiarism detection is the selection of suitable discriminators that reflect plagiarism and not co-incidental similarity due to content or theme. So far the focus has been on lexical and structural similarity in both program code and natural language, but these become less effective when the degree of rewriting or form of plagiarism becomes more complex. Three areas to address in automatic plagiarism detection include the following:

(1) Finding suitable discriminators of plagiarism which can be quantified.

(2) Developing suitable methods to compare those discriminators.

(3) Finding suitable measures of similarity.

The goal of an automatic plagiarism detection system is to assist manual detection by: reducing the amount of time spent comparing texts, making comparison between large numbers of multiple texts feasible and finding possible source texts from electronic resources available to the system. The systems must minimise the number of false positives (those incorrectly classed as plagiarised) and false negatives (those incorrectly classed as non-plagiarised), and maximize the number of true positives (those correctly classed as plagiarised) and true negatives (those correctly classed as non-plagiarised).

3.1. Plagiarism detection in program code

Parker and Hamblen (1989) define plagiarism in program code as: "a program which has been produced from another program with a small number of routine transformations." These transformations might range from the simple (e.g. changing comments or variable names), to the more complex (e.g. replacing control structures with equivalents). Faidhi and Robinson (1987) characterise possible transformations in a spectrum ranging from no change (level 1) to the most complex changes (level 6). Identifying all possible changes a plagiarist is likely to use to disguise their copying would be infeasible, however changes in program code tend to fall into two categories (Joy and Luck,1999):

(1) **Lexical changes**: edits that would, in principle, be performed using a simple text editor and require little knowledge of the programming language. In particular these edits would not require knowledge sufficient to parse a program. These changes might include rewording, adding or deleting comments, changing formatting and changing variable names (approximately levels 1-3 in Faidhi and Robinson (1987)).

(2) **Structural changes**: these changes require knowledge of the programming language to the extent of being able to change the structure so that the program still parses. This is highly language dependent and might include replacement of equivalent iteration structures or operand ordering (approximately levels 4-6 in Faidhi and Robinson (1987)).

Two major approaches have emerged for plagiarism detection in program code (e.g. (Whale,1990) and (Verco and Wise, 1996)): (1) **attribute-counting**, and (2) **structure-based**. The earliest systems (e.g. (Ottenstein,1976)) counted program attributes such as metrics of software science[5], the number of tokens, distribution of identifiers and other author or program specific characteristics. These scores were collected into a vector called a profile and a distance measure, e.g. Euclidean distance, used to determine the degree of closeness between profiles. Many attributes have been tried, but an attribute-counting method with consistent results is still to be found. The difficulty of choosing an appropriate set of metrics, so far, has been an insoluble problem associated with this approach.

Empirical investigation has shown that measures of similarity derived from attribute-counting measures cannot capture the structure of a program sufficiently to distinguish plagiarised from non-plagiarised texts (c.f. (Faidhi and Robinson,1987), (Whale,1990) and (Wise,1992)). This has led to the creation of 'structure-based' methods which compare string representations of program structure directly, rather than measures derived *from* the structure.

Programs are typically converted into a string representation, which might involve some knowledge of the target language, e.g. tokenisation, or more language-dependent methods, such as parsing. The aim of this stage is to reduce the effect of differences due to systematic changes

[5] "A rule for assigning a number or identifier to software, calculated algorithmically from the software alone" – (Dunsmore,1984).

such as renaming identifiers, or replacing equivalent iteration structures.

Given two "normalised" strings, a language-independent comparison method is chosen based upon string comparison, e.g. approximate string matching (Wise,1996) or sequence alignment (e.g. Gitchell and Tran (1999)), with the following assumption: plagiarised programs are more likely to share *longer matching substrings*. The output from this comparison can be quantitative, e.g. the longest common substring or the average substring length; or qualitative, such as a visual representation of comparison or a list of substring matches.

Some researchers combined both approaches, e.g. Donaldson et al. (1981) and Jankowitz (1988), but current detec-tion systems (see, e.g. Whale (1990), Baker (1993), Wise (1996), Gitchell and Tran (1999), Joy and Luck (1999) and Prechelt et al. (2000)) use only structure-based approaches and compare string representations of programs rather than metrics obtained from the program. Current research is not focused on finding further comparison methods, but in how best to recode programs into a normalised stream of tokens.

In summary, methods of detection between programs include measuring similarity: (1) between program style or complexity (Ottenstein,1976), (2) between their parse trees (Jankowitz,1988), (3) between their flow of data (Horwitz, 1990), (4) between fingerprint or compressed representations of the programs (see, e.g. (Manber,1994) and (Ziv and Lempel,1977)), (5) between matching substrings (e.g. the UNIX diff tool), (6) between compressed versions of the sources, (7) between parameterised matches (Baker,1993), and (8) between visual representations of the programs (Church and Helfman,1993).

3.2. Plagiarism detection in natural language

In a similar way to automatic plagiarism detection in program code, methods of plagiarism detection in natural language have focused on identifying similarity between groups of texts, rather than across a single text. However, natural language is inherently more difficult to process automatically than program code, making detection of even more routine changes such as lexical or syntactic modifications less successful.

Compared to program code, difficulty in processing in this instance is derived from at least two properties of natural language: (1) ambiguity, and (2) unconstrained vocabulary. For example, consider detecting plagiarism in which words are replaced by their synonyms. Machine Readable Dictionaries such as WordNet[6] are possible sources for such transformations, but because word senses are ambiguous, selection of the correct term is often non-trivial. Consider a more complex approach in which the texts are parsed and their syntax compared. Ambiguity in parsing means that the same sentence between texts can result in many different parse trees thereby reducing similarity and wrongly missing plagiarism.

The flexibility and complexity of natural language has driven researchers (in many language engineering tasks, not just plagiarism) to apply simpler methods of similarity involving a minimal amount of natural language processing. As with detection between different programs, various methods of comparing texts have been investigated, as well as defining suitable discriminators. Most research has been directed at finding possible copies and collusion between members of a closed-set, e.g. a set of student essays. However, unlike detection in software code, detection of plagiarism from Web-based sources has also been investigated using methods of copy detection.

Methods of detection originating from file comparison, information retrieval, authorship attribution, compression and copy detection have all been applied to the problem of plagiarism detection. For example, similarity between texts based on the longest common subsequence (e.g. UNIX diff), approximate string matching (e.g. turnitin.com), the overlap of longest common substrings (e.g. YAP (Wise,1996) and JPLAG (Prechelt et al.,2000)), the proportion of shared content words (particularly those occurring only once: the hapax legomena - CopyCatch (Woolls and Coulthard,1998)), the overlap of consecutive word seqences or word n-grams (e.g. Ferret (Lyon et al.,2001), SCAM (Shivakumar and Garcia-Molina,1996), COPS (Brin et al.,1995), Koala (Heintze,1996) and CopyFind[7]), and compressed versions of the texts (see, e.g. (Medorie et al.,2002))

Methods have also been developed to visualise the similarity between texts including VAST (Culwin and Lancaster,2001), Dotplot (Church and Helfman,1996), Bandit[8] and Duploc (Ducasse et al.,1999). For more information about plagiarism detection, see (Clough, 2000).

4. Example plagiarism detection methods for natural language

Two approaches for detecting plagiarism in natural language texts will now be discussed. The main focus of my research has been measuring text reuse in journalism. In most cases this would represent reuse by an experienced plagiarist (i.e. the newsworker) with a high level of editing skill and excellent command of the English language. More recently I have been experimenting with methods to detect stylistic inconsistency across a single text. This inconsistency may indicate that different authors may have been involved with its creation.

4.1. Measuring similarity between multiple texts

One method of plagiarism detection which has proven to be successful in a number of applications is finding the overlap of matching subsequences and substrings (consecutive tokens) of length ≥ *n* (where *n* is derived empirically).

[6] WordNet is a thesaurus of general English containing semantic relations between words and phrases. For more information see: http://www.cogsci.princeton.edu/~wn/ (site visited: 31/01/2003).

[7] http://plagiarism.phys.virginia.edu/software.html (site visited: 23/01/2003).

[8] http://socrates.cs.man.ac.uk/~ajw/pd.html (site visited: 23/01/2003).

Because there are many ways to express the same concept, the longer *n* becomes, the more unlikely it is that the same sequence of *n* tokens (words or characters) will appear in the same order in independently-written texts. Methods have varied between the use of fixed length substrings (commonly known as n-grams), variable-length substrings, or both (see, e.g. (Brin et al.,1995), (Shivakumar and Garcia-Molina,1996), (Lyon et al.,2001), and (Broder,1998)). The assumption that derived texts are unlikely to share matching sequences of consecutive tokens of length ≥ *n* rests upon the intuition that the way in which people express themselves varies from person to person. Even if writing about the same subject, it is unlikely that the topic will be expressed in *exactly* the same way (e.g. the same program structure or grammatical syntax), and using exactly the same words (i.e. the same vocabulary). As McEnery and Wilson (1996) state about finding the same sentence more than once: "unless it is a very formulaic sentence (such as those appearing as part of a legal disclaimer at the beginning of a book), it is deeply unlikely that you will find it repeated in its exact form in any book, in any library, anywhere" (McEnery and Wilson,1996:7).

For example, Table 1 shows the number of n-gram occurrences (tokens), distinct n-grams (types), the percentage of unique n-grams (type-token ratio) and the percentage of n-grams occurring in one text as *n* varies from 1 to 10 words. These figures are taken from 769 texts in the METER corpus (a restricted domain of law and court reporting texts - see (Clough et al.,2002b)). Note that as *n* increases, the percentage of distinct n-grams increases indicating that tokens become increasingly unique. Note also that the percentage of distinct n-grams occurring in a single file also increases indicating that longer n-grams are less likely to be shared across more than one document.

N (words)	N-gram occurrences (tokens)	Distinct n-grams (types)	% distinct n-grams	% distinct n-grams in 1 file
1	137204	14407	11	39
2	248819	99682	40	67
3	248819	180674	73	82
4	257312	214119	85	90
5	251429	226369	90	93
6	250956	231800	92	94
7	250306	234600	94	95
8	249584	236310	95	96
9	248841	237409	95	97
10	289610	278903	96	97

Table 1 Uniqueness of consecutive n-word sequences (n-grams) as *n* increases from 1-10 words

Although Table 1 shows that as *n* increases it is less likely that the n-word sequence will be repeated and therefore becomes a possible suitable discriminator for plagiarism, it also becomes less likely that the n-gram will occur at all in any derived text (a problem known as *data sparseness*). The selection of *n* is therefore a trade-off between correctly finding derived texts, versus the number of plagiarised texts retrieved.

Similarity between texts is measured by computing sets of n-grams for each text and comparing these to determine the degree of overlap. A similarity function is used to capture the degree of overlap between the two texts represented by the sets of n-grams and a threshold chosen above which texts are deemed plagiarised. We used a variety of different similarity measures to capture the overlap between two texts including the following:

$$\text{Similarity}(SA, SB) = \frac{\sum \text{leni} \times \log(\text{leni} + 1)}{|SA|} \quad i \in T$$

In this measure, the similarity between two sets of n-grams, S_A and S_B, is computed by summing elements of the intersection, T, of n-gram sets using a weighted function dependent on the length of n-gram i, len_i. The sum is normalized with respect to set S_A to create a subset or containment score. This score can be computed for various lengths of n (or len_i) where the weighted score is used to give a higher similarity score to longer overlapping n-grams. If the weighting based on length is not used, the problem arises where a single overlapping n-gram of length 5 will receive the same score as 5 n-grams of length 1, but where even a single occurrence of an n-gram of length 5 words is much more significant and indicative of reuse than many single word matches.

In my experiments with n-grams, I found data sparseness to be a problem: the occurrence of longer n-grams becomes increasingly rare in derived texts as relatively simple lexical and syntactic rewriting is used to modify the original. This method of comparison then becomes sensitive to the n-gram length chosen to discriminate between derived and non-derived texts. Another problem is the existence of long substrings in non-derived texts resulting from domain-specific terminology (e.g. "the court was told yesterday") and direct quotations, both of which are likely to occur in dependent and independently-written texts. An alternative approach to using fixed-length n-grams is variable length n-grams or sequence comparison. In particular, I have experimented with various methods of computing the overlap of *longest common substrings.*

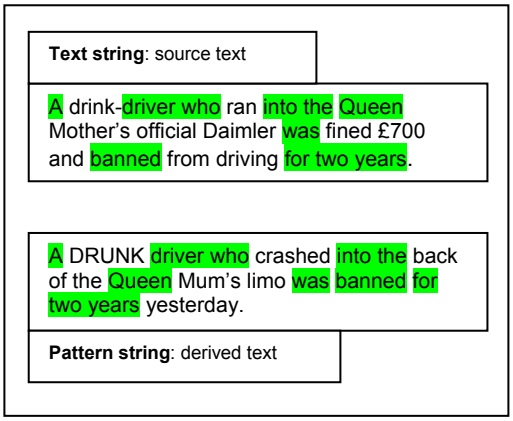

Figure 1 The longest common substrings computed between two sentences using GST

Various methods of sequence comparison have been suggested for measuring the similarity and difference between token sequences (see, e.g. (Sankoff and Krustal, 1983)). One approach used successfully for plagiarism detection in program code is Greedy String Tiling (or *GST*: see, e.g. (Wise,1993)), an algorithm which computes a 1:1 mapping between the tokens in a text pair in such a way that as much of one text as possible is covered with maximal non-overlapping substrings (called *tiles*) from the other. This algorithm computes the longest common substrings (greater than length *n*) between two texts without having to define an n-gram size *a priori*. For example, the texts in Figure 1 represents a tiling of two sentences after running GST (tiles are highlighted) with a minimum match length of 1 word.

The result of running the GST algorithm is a set of maximal matches between the text pair: [for two years], [driver who], [into the], [a], [queen], [was] and [banned]. Given a set of tiles, a number of different quantitative measures can be derived such as the minimum and maximum tile length, the average tile length, the dispersion of tile lengths, and a similarity score based on tile length (similar to that for n-gram containment). We have found on average that derived texts do share longer matching substrings, and both the tiling for a derived and non-derived text pair are in most cases apparently different (see Figure 2). The challenges are capturing these tiling patterns such that derived and non-derived texts are distinguishable.

Figure 2 The longest common substrings computed between a derived text pair (top) and independent text pair (bottom)

This method of comparison has been particularly useful in aiding manual comparison between texts[9]. The visualization of GST is similar to that of the Dotplot approach (see Appendix 1), but with the advantage that the similarity is also quantified. I am looking at ways to better capture tiling patterns between texts, as well as creating 1:1 mappings based on the context around tokens, rather than just selecting the first match. I am also experimenting with making tile creation more resilient to simple editing changes, and comparing alternative text representations based on tagged, parsed and interpreted formats.

I currently use a version of the UNIX diff tool to compute a measure of similarity based on the longest common *subsequence* of matching tiles to capture an element of structural similarity between the texts. This sequence comparison stage is also able to produce an *edit script* which lists a set of simple edit operations (insertion, deletion, substitution and transposition) to transform the original text into the derived text. My goal is to capture common editing patterns that describe how the tiles differ between the original and derived text, to enable the detection of further rewrite examples. Murata and Isaraha (2002) have shown how sequence comparison can be used to identify predictable differences between spoken transcripts of written texts.

For both n-gram and GST approaches, I am experimenting with methods to relax the matching between sequences to allow for: (1) small gaps to represent token deletion, (2) simple word substitution (using WordNet), (3) the insertion of certain words such as domain-specific terminology and function words (e.g. conjunctions), and (4) simple re-ordering of tokens (e.g. transposition). These are aimed at preserving longer matching n-grams and tile lengths, and making the approaches resilient to "simple" edits.

4.2. Identifying inconsistencies within a single text

Given a natural language text, consider the task of identifying inconsistencies within it. These might be stylistic (e.g. use of particular words), grammatical (e.g. use of punctuation) or other forms specific to plagiarism detection, e.g. inconsistent use of references. If we concentrate on identifying stylistic and grammatical inconsistencies, this problem is similar to two areas: (1) authorship attribution (see, e.g. (Holmes,1994)), and (2) detecting and correcting inconsistencies in collaborative writing (see, e.g. (Glover and Hirst,1995)). The latter is particularly interesting as this task first involves identifying inconsistencies before recommendations can be made to correct the style. Typical indicators of style include average sentence length, distribution of word classes, verb forms, nominal forms (e.g. gerunds), vocabulary richness and frequency of passive voice. Indicators of style can be derived from un-analysed text, tagged text, parsed text or interpreted text.

One technique I have been exploring is the *cusum* technique (Farringdon,1996). This was developed to detect stylistic inconsistencies through variation in the proportion of occurrences of author-specific discriminators called *habits*. The technique has been used in British courts as evidence to prove or disprove authorship, e.g. witness

[9] This implementation of the GST algorithm can be tested on-line at: http://nlp.shef.ac.uk/meter/index.jsp (site visited 31/01/2003).

statements and suicide notes. Over the years this approach has received considerable criticism from academics, due to its rather *ad hoc* underlying assumptions (see, e.g. Holmes and Tweedie (1995)). However there is evidence to suggest that this method does work well in some cases, and with certain changes can be made more reliable (see, e.g. (Somers,1998)). The question is whether or not the cusum approach can be used in plagiarism detection.

The underlying assumption of the technique is that everyone has a unique set of quantifiable habits (this assumption has received the most criticism), e.g. noun usage, use of 2-3 letter words, words beginning with a vowel and use of function words. The claim is that when compared with the average sentence length, the rate of habit occurrence will be consistent. Any variation between the habit and average sentence length supposedly indicates multiple authorship.

The habits and average sentence length are plotted on the same chart known as a cumulative sum chart that plots the *cumulative* deviation from the mean. This is not just a plot of separate deviation scores for each sentence, but rather a cumulative measure of homogeneity. Given the number of words in sentence r, w_r, for sentences $r = 1 \ldots n$, the average sentence length, $\overline{w}$, is given by:

$$\overline{w} = \frac{1}{n}\sum_{r=1}^{n} w_r$$

The variance of each sentence is computed and summed with those preceding it. For each sentence, *i,* the cusum value, c_i, is given by:

$$c_i = \sum_{r=1}^{i} (w_r - \overline{w})$$

The same calculation is performed for the habits, but first computing the average number of habit occurrences per sentence, before then computing the habit variance for each sentence, in a cumulative fashion. After plotting the average sentence length and habits, the axes are scaled such that they lie within the same range, making the charts comparable.

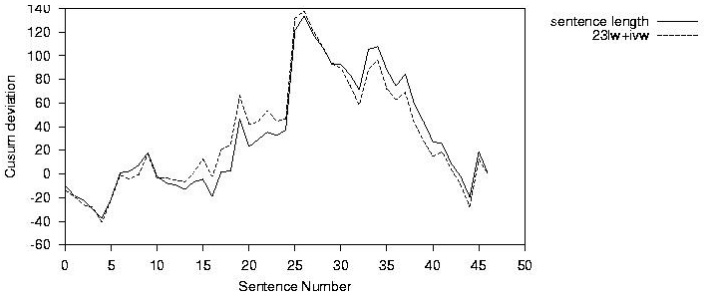

Figure 3 Example cusum chart for Chapter 1 of Jane Austin's Northanger Abbey

To scale the charts, I use ordinary least squares regression between the sets of values for the habit and average sentence length. This computes a linear equation of the form $y=ax+b$ by minimizing the differences between the two data sets. I then re-scale the habit values using this linear equation.

Figure 3 shows an example cusum plot for average sentence length versus the habit: 2-3 letter words + words starting with an initial vowel word for Chapter 1 of Northanger Abbey by Jane Austin[10]. This cusum chart represents one of the difficulties of using cusums to identify inconsistencies in texts: that of selecting the correct habits. The charts can be seen to deviate from around sentence 16 to 23, and 32 to 42, which would suggest Jane Austin was not the only author of this chapter. However, for the BBC news story given in Figure 4, this choice of habit does seem suitable (assuming the text is authored by one person).

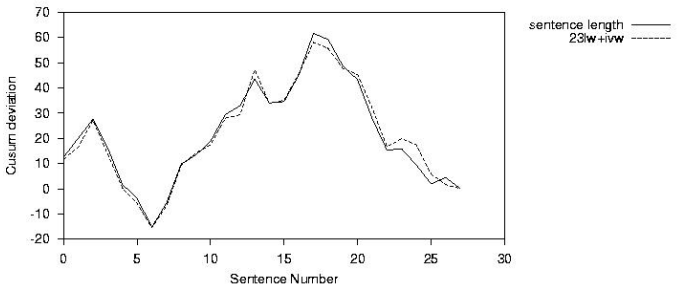

Figure 4 BBC news story

Figure 5 shows two news stories (about the same topic) from two different news organisations appended together at sentence 11 (vertical line). Again, like Figure 3, the choice of habit is not suitable in distinguishing the two authors and there seems little consistency between the texts, making it difficult to conclude anything substantive from these charts.

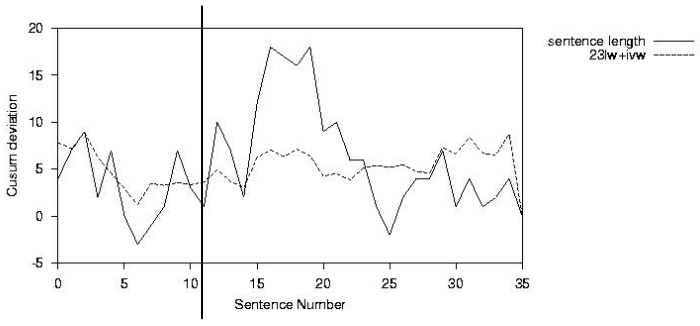

Figure 5 Two combined news stories (from the Sun and Mirror – British tabloids)

[10] Electronic versions of many literary classics can be found on the Project Gutenberg web site. http://promo.net/pg/ (site visited: 31/01/2003)

There are three main problems with using cusums: (1) computing the scaling factor for plotting cusum charts, (2) selecting suitable habits, and (3) detecting "significant" variations between the charts. Further problems also involve the quality of sentence splitting, the removal of "noisy" sentences, e.g. lists which deviate significantly from the average sentence length, the chart's appearance varies depending on the number of sentences analysed, combining texts in different ways tends to give different cusum charts and the position of the combined texts also creates variation in the cusum plots. The cusum technique is certainly useful in detecting inconsistencies within a single text, but work by Hearsee (2001) found the method unsuccessful for plagiarism detection, and I believe methods used to detect inconsistencies in collaborative writing may offer better alternatives to the cusum approach.

5. Recommendations for future work

Many academics and commercial organisations have addressed the problem of plagiarism detection and many areas are still part of active research among both of these communities. There are still many questions surrounding plagiarism and its detection which remain unanswered, but I would like to recommend at least five areas of research that I believe would benefit plagiarism detection. These recommendations focus on detection in natural language texts, because to some degree the problem for program code has already been solved.

5.1. Multi-lingual detection

Research in natural language has predominantly focused on plagiarism occurring between monolingual texts (mainly English). However, given the multi-lingual nature of the Web and access to a growing wealth of multi-lingual resources, it seems plausible that plagiarists (who are probably able to write in more than one language) will reuse texts from a variety of languages as well as sources to create new texts.

In the near future, it may well be common to find derived texts in one language, but with sources existing only in another language. Without knowing the source language from which the derived text has been taken, it becomes almost impossible to find possible source texts even with Cross Language Information Retrieval (CLIR) and Multilingual Copy Detection (MCD) systems. Another approach is to use bilingual alignment techniques to find the most likely corresponding source texts, but these approaches often only work between languages which share similar orthographic word forms (known as *cognates*), e.g. some proper names in German and English.

5.2. A text collection for plagiarism detection

Being able to study plagiarism and compare various approaches to plagiarism detection is essential to further research in this area. To date, no standard collection of texts for plagiarism detection in natural language exists, thereby making comparison between various approaches impossible, unless the same set of texts is used.

Many areas of Language Engineering, including Information Retrieval (IR), summarisation, document routing, genre classification, authorship analysis, and information extraction have benefited from careful construction of a standard evaluation test collection. For example, in IR the Text REtrieval Conference (TREC) has provided document collections, methods of evaluation and problem-specific resources such as lists of user requests (topics) and documents judged as relevant to those topics for many years. The TREC collections are considered to be a driving force behind much of the success in recent years of IR evaluation and comparison. I believe that building a test collection for plagiarism detection would offer many benefits including the following:

(1) It would help to stimulate research in automatic plagiarism detection.

(2) It would enable communities to compare different approaches.

(3) It would help us better understand plagiarism.

(4) It could be used to help teach students how to cite and paraphrase correctly by looking at examples of plagiarism.

However, building a representative and comprehensive test collection is unlikely to be straightforward. One of the issues which most builders of corpora face is copyright. In plagiarism this is no exception. Either examples used in the collection must be created artificially (not the ideal), or steps to protect confidentiality must be taken (e.g. sanitation). A possible solution is to use corpora created for other tasks involving text reuse, e.g. summarisation, translation and journalistic text reuse, but to be of any use, these must provide realistic examples of plagiarism.

5.3. Use of natural language processing

Most approaches used in plagiarism detection so far have involved minimal natural language processing (NLP) because plagiarism can involve complex editing that even sophisticated methods of analysis are probably unable to deal with. However, there are many areas of NLP that could aid plagiarism detection, particularly in identifying texts which exhibit similarity in semantics, structure or discourse, but differ in lexical overlap and syntax.

These may include morphological analysis, part-of-speech tagging, anaphora resolution, parsing (syntactic and semantic), co-reference resolution, word sense disambiguation and discourse processing. I believe that future work would benefit from exploring these areas for detecting similarity to perhaps offer several similarity scores based on lexical overlap, syntax, semantics, discourse and maybe other structural features.

5.4. Use of techniques from machine learning

Given more sophisticated methods of analysis, more sophisticated methods of combining evidence from these sources would benefit plagiarism detection. I have used a Naïve Bayes probabilistic classifier to combine evidence

from several measures of similarity taken from a GST tiling and make a decision: derived or not-derived, with a reasonable degree of success. This involves using texts which have already been classified as plagiarised or not to training the classifier (called *supervised learning*), but other methods of learning, e.g. *unsupervised learning,* can also be helpful in grouping together texts which exhibit similar characteristics (e.g. *clustering*).

5.5. Detection within single texts

Detecting inconsistencies within a single text is a challenging problem and I have illustrated some of the difficulties with one technique: the cusum. However, given that studies based on interviews with academic staff have shown that finding stylistic inconsistencies through manual inspection of texts often reveals plagiarism, it would seem this form of plagiarism detection deserves more attention. There are many approaches to authorship attribution and style analysis, including those from computational stylometry which may offer more suitable solutions to this problem than the cusum. These could also be explored.

The second challenge for the single text is to extract the inconsistencies and then find possible source texts. After all, it is not much use to have these inconsistencies without being able to compare them with possible original source texts. This presents the challenge of best extracting the inconsistencies (e.g. select sentences or phrases), and then formulating suitable search requests against on-line collections.

6. Conclusions

In this paper I have considered the problem of plagiarism, one of the most publicised forms of text reuse around us today. In particular, I have focused on automatic plagiarism detection, the task of identifying quantifiable discriminators able to distinguish derived from non-derived texts. I have discussed various approaches to plagiarism detection in both program code and natural language. To date there are few resources which specifically address the task of automatic plagiarism detection. This paper outlines many of the approaches taken over the past 30 years.

The use of automatic methods of detection aids the manual inspection of suspect texts by reducing the effort required in comparing large numbers of texts, and finding possible sources from on-line collections. Automatic detection is used to bring to light texts which exhibit high similarity with others, or high inconsistency within them and almost all approaches assume that a high degree of similarity based on quantifiable discriminators (lexical or structural) would not be likely from texts written independently by a single author.

Structure-based methods of plagiarism detection have proven successful for program code. The task in their case is not so much one of finding methods of comparing programs, but rather finding suitable representations that are not affected by both trivial and more complex transformations. The majority of work has focused on detecting plagiarism or collusion between a group of programs, rather than finding possible on-line sources.

The task of plagiarism detection in natural language is more complex than with program code because of ambiguity and the wide variety of ways in which the same topic can be expressed. Most techniques have concentrated on finding unlikely structural patterns or vocabulary overlap between texts, finding texts from large collections (copy detection) and collaboration between texts. I have presented a number of methods for plagiarism detection that involve minimal Natural Language Processing. Due to the difficulty in processing natural language, these methods are likely to offer more robust solutions across varying domains and languages. However without using more complex language analysis, cases of plagiarism may be missed.

I believe that plagiarism detection offers an interesting and challenging problem for researchers from a variety of backgrounds and will continue to be of interest as cases of plagiarism are publicised and motivation intensified by the concerns of authors, both academic and commercial.

7. Notes

a As an example of the last, rather extreme action, consider the legal action taken by Ordnance Survey against the Automobile Association (AA) for copy right infringement which ended in 2001 after a long running court battle. In the end, AA admitted using Ordnance Survey originals as the basis for their own maps which were then published; not permissible in the copyright agreement agreed with Ordnance Survey. To protect their intellectual property, Ordnance Survey admitted they introduced minor deliberate “errors” into their maps (e.g. slight changes to the bend in a river) to enable them to prove copyright infringement had occurred.

8. References

Angélil-Carter, S. (2000), Stolen Language - plagiarism in writing, Real Language Series, Pearson Education Limited.

Baker, B. (1993), Parameterised Pattern Matching: Algorithms and Applications, Proceedings of the 25th Annual ACM Symposium on Theory of Computing, ACM Press, 71-80.

Bell, A. (1991), The Language of News Media, Blackwell.

Bell, A. (1996), Text, time and technology in News English, In Re-designing English: news texts, new identities, Routledge, 3-26.

Brin, S., Davis, J. and Garcia-Molina, H. (1995), Copy Detection Mechanisms for Digital Documents, Proceedings of the ACM SIGMOD International Conference on Management of Data, 398-409.

Broder, A. Z. (1998), On the resemblance and containment of documents, Compression and Complexity of Sequences, IEEE Computer Society.

Bull, J., Collins, C., Coughlin, E. and Sharp, D. (2001), Technical Review of Plagiarism Detection Software Report, Computer-Assisted Assessment Centre, University of Luton.

Church, K.W. and Helfman, J.I. (1993), Dotplot: A Program for Exploring Self-Similarity in Millions of Lines of Text and Code, Journal of Computational and Graphical Statistics, Vol. 2(2), 153-174.

Clough, P.D. (2000), Plagiarism in Natural and Programming Languages: An Overview of Current Tools and Technologies, Department of Computer Science, University of Sheffield, UK, Technical Report CS-00-05.

Clough, P.D., Gaizauskas, R., Piao, S.L. and Wilks, Y. (2002a), Measuring Text Reuse, *In Proceedings of the 40th Anniversary Meeting for the Association for Computational Linguistics*.

Clough, P.D., Gaizauskas, R. and Piao, S.L. (2002b), Building and annotating a corpus for the study of journalistic text reuse, Proceedings of the 3rd International Conference on Language Resources and Evaluation, LREC2002, 1678-1685.

Clough, P.D. (2003), Measuring Text Reuse, PhD thesis, University of Sheffield.

Culwin, F. and Lancaster, T. (2000), A review of electronic services for plagiarism detection in student submissions, Proceedings of 8th Annual Conference on the Teaching of Computing.

Culwin F. and Lancaster T. (2001), Visualising Intra-Corpal Plagiarism, Presented at Information Visualisation 2001, London, UK.

Donaldson, J. L., Lancaster, A. and Sposato, P. H. (1981), A Plagiarism Detection System, ACM SIGSCI Bulletin, Vol. 13(1), 21-25.

Ducasse, S. and Rieger, M. and Demeyer, S. (1999), A Language Independent Approach for Detecting Duplicated Code, Proceedings ICSM'99 (International Conference on Software Maintenance), IEEE, 109-118.

Dunsmore, H. E. (1984), Software Metrics: An Overview of an Evolving Methodology, Information Processing and Management, Pergamon Press Ltd., Vol. 20(1-2), 183-192.

Farringdon, J. M. (1996), Analysing for Authorship: A Guide to the Cusum Technique. Cardiff: University of Wales Press.

Faidhi, J. A. W. and Robinson, S. K. (1987), An Empirical Approach for Detecting Program Similarity and Plagiarism within a University Programming Environment, Journal of Computer Education, Pergamon Journals Ltd., Vol. 11(1), 11-19,

Gitchell, D. and Tran, N. (1999), Sim: A utility for Detecting Similarity in Computer Programs, Proceedings of 13th SIGSCI Technical Symposium on Computer Science Education, 226-270.

Glover, A. and Hirst, G. (1995), Detecting stylistic inconsistencies in collaborative writing, In Thea van der Geest et al., editor, Writers at work: Professional writing in the computerized environment, Springer, London.

Hannabuss, S. (2001), Contested texts: issues of plagiarism, Library Management, MCB University Press, Vol. 22(6-7), 311-318.

Heintze, N. 1996), Scalable Document Fingerprinting, In Proceedings of the Second USENIX Workshop on Electronic Commerce.

Helfman, J.I. (1996), Dotplot Patterns: A Literal Look at Pattern Languages, In Theory and Practice of Object Systems (TAPOS) special issue on Patterns, Vol. 2(1), 31-41.

Hersee, M. (2001), Automatic Detection of Plagiarism, 3rd Year undergraduate project, University of Sheffield.

Hislop, G. W. (1998), Analyzing existing software for software reuse, Journal of Systems and Software, Vol. 41, 33-40.

Holmes, D. I. (1994), Authorship Attribution, Computers and the Humanities, Vol. 28(2), 87-106.

Holmes, D. I. and Tweedie, F. J. (1995), Forensic Linguistics: A review of the Cusum controversy. Revue Informatique et Statistique dans les Sciences Humaines. Vol. 31, 19-47.

Horwitz, S. (1990), Identifying the Semantic and Textual Differences Between Two Versions of a Program, Proceedings of Conference on Programming Language Design and Implementation (SIGPLAN'90), 234-245.

Jankowitz, H. T. (1988), Detecting plagiarism in student Pascal programs, Computing Journal, Vol. 31(1), 1-8.

Joy, M. and Luck, M. (1999), Plagiarism in Programming Assignments, IEEE Transactions of Education, Vol. 42(2), 129-133.

LaFollette, M. C. (1992), Stealing into Print: Fraud, Plagiarism, and Misconduct in Scientific Publishing. Berkeley: University of Calfornia Press.

Lyon, C., Malcolm, J. and Dickerson, B. (2001), Detecting Short Passages of Similar Text in Large Document Collections, In Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing, 118-125.

Mallon, T. (1989), Stolen Words: Forays into the Origins and Ravages of Plagiarism, Ticknor and Fields.

Manber, U. (1994), Finding similar files in a large file system, Proceedings of 1994 Winter Usenix Technical Conference, 1-10.

Martin, B. (1994), Plagiarism: a misplaced emphasis, Journal of Information Ethics, Vol. 3(2), 36-47.

McEnery, A. M. and Wilson, A. (1996), Corpus Linguistics, Edinburgh textbooks in empirical linguistics.

Medori, J., Atwell, E., Gent, P. and Souter, C. (2002), Customising a Copying-Identifier for Biomedical Science Student Reports: Comparing Simple and Smart Analyses, M. O'Neill et al. (Eds), AICS2002, LNAI 2464, Springer-Verlag, 228-233..

Murata, M. and Isahara, H. (2002), Automatic Extraction of Differences between Spoken and Written Languages, and Automatic Translation from the Written to the Spoken Language, Proceedings of the 3rd International Conference on Language Resources and Evaluation, LREC2002.

Osen, J. (1997), The Cream of Other Men's Wilt: Plagiarism and Misappropriation in Cyberspace, Computer Fraud and Security, Elsevier Science Ltd, 13-9.

Ottenstein, K. J. (1976), An algorithmic approach to the detection and prevention of plagiarism, SIGSCI Bulletin, Vol. 8 (Part 4), 30-41.

Parker, A. and Hamblen, J. O. (1989), Computer Algorithms for Plagiarism Detection, IEEE Transactions on Education, Vol. 32(2), 94-99.

Prechelt, L., Malpohl, G. and Philippsen, M. (2000), JPlag: Finding plagiarisms among a set of programs, Faculty of Informatics, University of Karlsruhe, Technical Report 2000-1.

Samuelson, P. (1994), Self-Plagiarism or Fair Use?, Communications of the ACM, Vol. 37(8), 21-25.

Sankoff, D. and Kruskal, J. (1983), Time Warps, String Edits, and Macromodules: The Theory and Practice of Sequence Comparison, Addison-Wesley.

Shivakumar, N. and Garcia-Molina, H. (1996), Building a Scalable and Accurate Copy Detection Mechanism, Proceedings of 1st ACM Conference on Digital Libraries DL'96.

Somers, H. (1998), An Attempt to Use Weighted Cusums to Identify Sublanguages, In Proceedings of the Joint Conference on New Methods in Language Processing and Computational Natural Language Learning, NeMLaP3/CoNLL98, 131-139.

Verco, K.L. and Wise, M., (1996), Software for Detecting Suspected Plagiarism: Comparing Structure and Attribute-Counting Systems, Presented at the First Australian Conference on Computer Science Education, Sydney, Australia, 130-134.

Whale, G. (1990), Identification of Program Similarity in Large Populations, The Computer Journal, Vol. 33(2), 140-146.

Wise, M. (1992), Detection of Similarities in Student Programs: YAP'ing may be preferable to Plague'ing, Presented at 23rd SIGCSE Technical Symposium, Kansas City, USA, 268-271.

Wise, M. (1993), Running Karp-Rabin Matching and Greedy String Tiling, Basser Department of Computer Science, Sydney University, Technical Report 463.

Wise, M. (1996), YAP3: Improved Detection of Similarities in Computer Programs and Other Texts, Presented at SIGCSE'96, 130-134.

Woolls, D. and Coulthard, M. (1998), Tools for the Trade, Forensic Linguistics, Vol. 5(1), 33-57.

Ziv, J. and Lempel, A. (1977), A universal algorithm for sequential data compression, IEEE Transactions Information Theory, Vol. 23, 337-343.

9. Appendices

Appendix 1 *Comparing texts using Dotplots*

Given two sequences, it is useful to be able to visualise the similarity between them to highlight regions of similar *structure*. The Dotplot is one such method which has been used to identify similarity between biological sequences (e.g. DNA and protein), to highlight plagiarism between both natural and programming language texts, to identify duplication and redundancy in software code, and as a preliminary stage for the alignment of bilingual texts. The Dotplot is a language-independent method of comparing two sequences because it can be used for any kind of sequence and across a variety of domains. The beauty of this approach is its simplicity but effectiveness as it relies one the human interpretation of similarity patterns.

A pre-processing stage is used to create the two token streams to compare. For texts this can be either groups of overlapping character n-grams, words, lines, sentences, or groups of overlapping word n-grams. Given two sequences of tokens, each token in one sequence is compared to every token in the other sequence using an exact or inexact match method. To visualize the similarity between a text pair, a dot is placed on a 2-by-2 matrix whenever the tokens match using either a single colour for an exact match, or a shade to indicate the degree of match. For example if the 3rd token of one text matches the 5th in another, a dot is placed at position (3,5) in the matrix and this is continued until all tokens have been processed. The result of the Dotplot is a visualization of matches between the two sequences where *diagonal lines* indicate *ordered* matching sequences, and *squares* indicate *unordered* matches.

Figure 6 illustrates the concept using a well-known phrase comparing words which match exactly. The left-hand image demonstrates self-similarity by comparing the same phrase which produces the main diagonal line. The smaller diagonals are due to repetition within the phrase of “to be”. The right-hand image demonstrates the effects of re-ordering one version such that the beginning is almost the same, but the end is different.

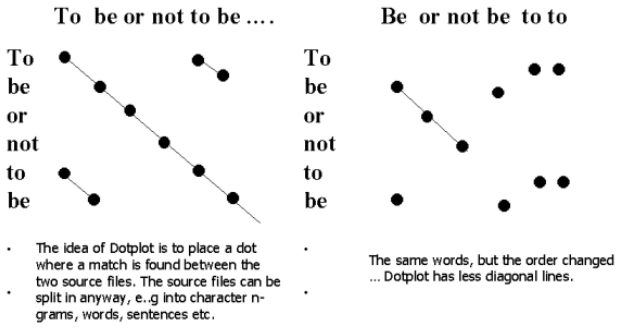

Figure 6 A simple Dotplot example

Figure 7 shows four patterns which can be created by the repetition of subsequences within the sequences themselves.

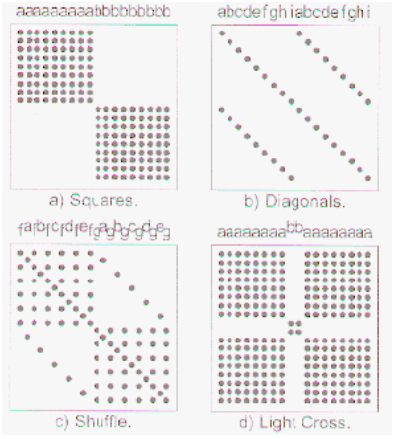

Figure 7 Four example patterns using a Dotplot visualization (Helfman,1996)

The Dotplot can be used for two purposes: (1) to identify regions of duplication *within* a text, and (2) to identify regions of similarity *across* texts. The Dotplot not only

shows the matches, but also the position at which they occur, therefore providing an indication of structural similarity.

To demonstrate how useful a visual representation of the similarity between texts can be, consider Figure 8. This shows two revisions of an academic paper (in Latex): the first (on the vertical axis) is my version of the paper; the second (on the horizontal axis) the result of a second author rewriting the paper to give it more impact and clarity. The Dotplot shows matching sequences of 20 characters (this removes most of the "noisy" text).

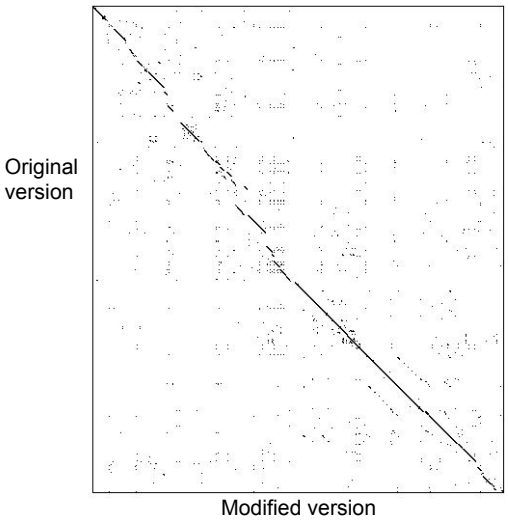

Figure 8 A comparison of a paper and its revision showing the location of most modification (in the earlier sections)

There are several points to notice about the resulting Dotplot pattern and what this says about the two texts:

(1) There is a noticeable line down the main diagonal indicating the second text is a clear revision of the first.
(2) Most revision was undertaken in the first half of the paper. There was good reason for this: the latter half of the paper contained the experiments and discussion which was on the whole considered acceptable. The main rewriting was in the introduction and background to improve clarity and style.
(3) The main diagonal at the end of the second half is broken towards the end. This represents a subsequent revision in the conclusions reflecting the changes made in the first half of the paper.
(4) The main diagonal is offset at certain positions. This represents the deletion of text in the original which results in a shorter text overall.
(5) Given the match length of 20 characters, the resulting matches indicate text reuse.
(6) Matches elsewhere between the texts are mostly due to Latex commands.
(7) Some of the gaps in the lines are caused by re-ordering paragraphs in the original version.

Appendix 2 *Cusum charts for chapters 1-5 of Jane Austin's Northanger Abbey*